Una-al-mes: Episodio#1 – Parte I - Hispasec

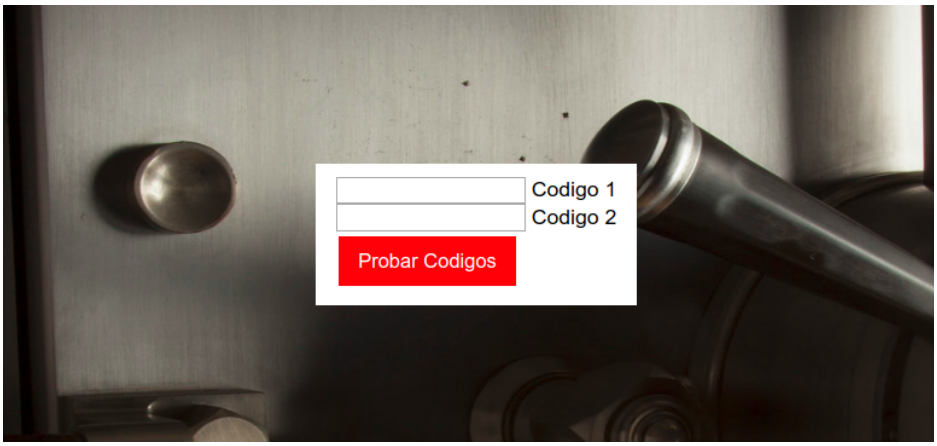
Enunciado CTF:

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 ;).

Resolución:

Accedemos a la URL que nos facilita el reto:
http://34.253.233.243/lacasadepapel/episodio1/puerta.php

Tenemos un formulario en el que nos piden dos códigos y un archivo ZIP protegido con contraseña :



Si analizamos el código fuente de "puerta.php" pulsando F12 en Chrome vemos que tiene un comentario que nos sugiere el uso de unos códigos:

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

Probamos 1234/1234 y admin/admin sin resultados positivos.

En la pestaña "Sources" podemos ver todo lo que se descarga al hacer la petición a la página "puerta.php". Vemos que hay un archivo javascript llamado "login.js" con el siguiente código comentado:

```
function conexion(){
    var Password = "unescape%28String.fromCharCode%252880%252C%2520108%252C%252097%252C%2520110%2529%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];
    }
}

}
```

Analizando el código vemos que guarda una cadena de texto en una variable llamada "Password" y que luego la "splitea" o la divide en dos substrings delimitados por el carácter ":".

Cada uno de ellos los guarda en dos variables, quedando de la siguiente forma:

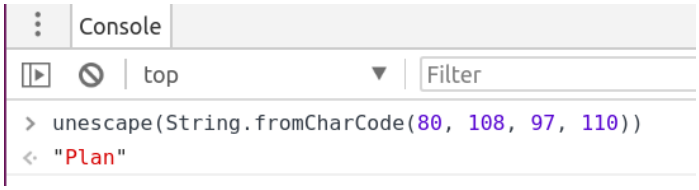
code1 = unescape%28String.fromCharCode%252880%252C%2520108%252C%252097%252C%2520110%2529%29
code2 = KZQWYZLOMNUWC===

Obtener *code1*:

Los símbolos de "%" nos indican que se trata de una codificación URL. Si lo decodificamos un par de veces usando un decodificador online, obtenemos un comando javascript:

- Cadena original: `unescape%28String.fromCharCode%252880%252C%2520108%252C%252097%252C%2520110%2529%29`
- Primer URL decoder: `unescape(String.fromCharCode%2880%2C%20108%2C%2097%2C%20110%29)`
- Segundo URL decoder: `unescape(String.fromCharCode(80, 108, 97, 110))`

Ejecutamos este código en el sandbox de Chrome, para ello pulsamos "F12" y con "Esc" podemos introducir código javascript para poder ser ejecutado, obteniendo el valor del primer código: **Plan**



Obtener *code2*:

La cadena de texto contiene los símbolos "=" por lo que nos puede dar la pista de que se trata de un base64 o base32. Probamos a decodificarlos a través de la terminal de Linux:

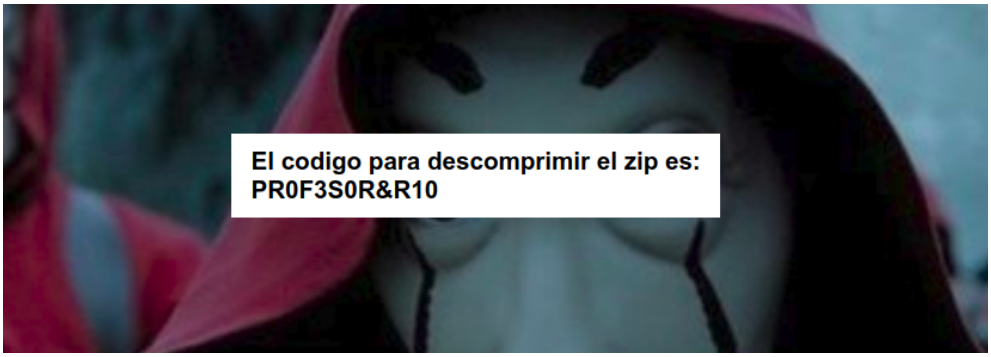
Con base64 no obtenemos nada:

```
$ echo "KZQWYZLOMNUWC===" | base64 -d
)@#a@@0@#base64: entrada inválida
```

Con base32 obtenemos el valor del code2, cuyo valor es **Valencia**:

```
$ echo "KZQWYZLOMNUWC===" | base32 -d
Valencia
```

Introduciendo ambos códigos en el formulario y pulsando el botón de "Probar códigos", obtenemos la contraseña del archivo ZIP para poder descomprimirlo:



Dentro del archivo comprimido hay un ejecutable de Windows llamado "episodio1.exe", que al ejecutarlo con "wine" en Linux, nos da la siguiente salida:

```
$ wine episodio1.exe
System_Date: 05/18/18
Wrong date R3m0!
-------------HINT---------------------
'La persistencia de la memoria...'
----------------------------------
Pulse cualquier tecla para continuar...
```

Nos da un error indicando que la fecha del sistema es inválida. Tendremos que averiguar cuál tiene que ser la fecha válida del sistema para que la ejecución no nos de el error.

Lo podemos resolver de dos formas distintas:
A) Cambiando la fecha del sistema.
B) Editando el código del archivo con radare2.

Solución A: Cambiar la hora del sistema.

Si abrimos el archivo con un editor hexadecimal como Bless o si le aplicamos el comando “strings” con un “grep” buscando el mensaje de error, podemos ver una fecha:

```
$ strings episodio1.exe | grep "R3m0" -A 7 -B 7
01/23/89
Congratulation!!, Stealing Money $$$...
--------------------------
Stolen: 1.000.000.000 $
Flag:
.......................................
System_Date:
Wrong date R3m0!
-------------HINT---------------------
'La persistencia de la memoria...'
----------------------------------
```

Vemos que nos da la fecha 01/23/89 pero no aparece el valor del flag.

Si cambiamos la fecha del sistema operativo, cambiando la configuración de la fecha y la hora para poner la que nos pide el archivo, conseguimos una ejecución satisfactoria.

Con el comando “date” podemos comprobar que se ha cambiado bien la fecha del sistema una vez hemos accedido al panel de configuración y hemos hecho los cambios (en mi caso es Ubuntu):

Antes del cambio:

```
$ date
vie may 18 16:14:19 CEST 2018
```

Después del cambio:

```
$ date
lun ene 23 01:00:24 CET 1989
```

Si ejecutamos el archivo obtenemos el valor del hash md5 para el flag:

```
wine episodio1.exe
Congratulation!!, Stealing Money $$$...
--------------------------
Stolen: 1.000.000.000 $
--------------------------
Flag: e30f35ad8d9cb6efc0778539a669fa85
.......................................
Pulse cualquier tecla para continuar...
```

Por lo tanto el flag es **UAM{e30f35ad8d9cb6efc0778539a669fa85}**

Solución B: Cambiar el código del archivo con radare2

Vamos a analizar el archivo con la herramienta radare2. Para ello, abrimos el archivo en modo lectura:

```
$ r2 -w episodio1.exe
```

Analizamos el archivo:

```
[0x00401500]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Use -AA or aaaa to perform additional experimental analysis.
[x] Constructing a function name for fcn.* and sym.func.* functions (aan)
```

Entramos en el modo visual pulsando “V” y analizamos su ejecución.

A partir de la dirección “0x004015f2” es la parte del código que se ejecutará para devolvernos el flag, lo sabemos porque vemos el string “Congratulation”:

```
0x004015ec      0f8485000000    je 0x401677                 ;[5]
0x004015f2      488d15577c08.   lea rdx, str.Congratulation    Stealing Money    ...    ; 0x489250 ; "\nCongratulation!
0x004015f9      488b0d50c408.   mov rcx, qword [0x0048da50]    ; [0x48da50:8]=0x487a00
0x00401600      e87be60600      call sym.std::basic_ostream_char_std::char_traits_char___std::operator___std::char_trait
0x00401605      488d156d7c08.   lea rdx, str.                 ; 0x489279 ; "\n----------------------------"
0x0040160c      488b0d3dc408.   mov rcx, qword [0x0048da50]    ; [0x48da50:8]=0x487a00
0x00401613      e868e60600      call sym.std::basic_ostream_char_std::char_traits_char___std::operator___std::char_trait
0x00401618      488d15787c08.   lea rdx, str.Stolen:_1.000.000.000    ; 0x489297 ; "\nStolen: 1.000.000.000 $ "
0x0040161f      488b0d2ac408.   mov rcx, qword [0x0048da50]    ; [0x48da50:8]=0x487a00
0x00401626      e855e60600      call sym.std::basic_ostream_char_std::char_traits_char___std::operator___std::char_trait
0x0040162b      488d15477c08.   lea rdx, str.                 ; 0x489279 ; "\n----------------------------"
0x00401632      488b0d17c408.   mov rcx, qword [0x0048da50]    ; [0x48da50:8]=0x487a00
0x00401639      e842e60600      call sym.std::basic_ostream_char_std::char_traits_char___std::operator___std::char_trait
0x0040163e      488d156c7c08.   lea rdx, str.Flag:            ; 0x4892b1 ; "\nFlag: "
0x00401645      488b0d04c408.   mov rcx, qword [0x0048da50]    ; [0x48da50:8]=0x487a00
0x0040164c      e82fe60600      call sym.std::basic_ostream_char_std::char_traits_char___std::operator___std::char_trait
0x00401651      4889c1          mov rcx, rax
```

Sin embargo, a partir de la dirección “0x00401677” podemos ver la parte del código que se ejecutará si la fecha del sistema es errónea:

```
0x00401672      e994000000      jmp 0x40170b
; CODE XREF from 0x004015ec (sym.main)
0x00401677      488d15707c08.   lea rdx, str.System_Date:    ; 0x4892ee ; "System Date: "
0x0040167e      488b0dcbc308.   mov rcx, qword [0x0048da50] ; [0x48da50:8]=0x487a00
0x00401685      e8f6e50600      call sym.std::basic_ostream_char_std::char_traits_char___std::operato
char___std::basic_ostream_char_std::char_traits_char____charconst
0x0040168a      4889c1          mov rcx, rax
0x0040168d      488d4510        lea rax, [arg_10h]          ; 0x10 ; 16
0x00401691      4889c2          mov rdx, rax
0x00401694      e8e7e50600      call sym.std::basic_ostream_char_std::char_traits_char___std::operato
char___std::basic_ostream_char_std::char_traits_char____charconst
0x00401699      488d155c7c08.   lea rdx, str.Wrong_date_R3m0 ; 0x4892fc ; "\nWrong date R3m0!"
0x004016a0      488b0da9c308.   mov rcx, qword [0x0048da50] ; [0x48da50:8]=0x487a00
0x004016a7      e8d4e50600      call sym.std::basic_ostream_char_std::char_traits_char___std::operato
char___std::basic_ostream_char_std::char_traits_char____charconst
0x004016ac      488d155b7c08.   lea rdx, [0x0048930e]        ; "\n"
0x004016b3      488b0d96c308.   mov rcx, qword [0x0048da50] ; [0x48da50:8]=0x487a00
0x004016ba      e8c1e50600      call sym.std::basic_ostream_char_std::char_traits_char___std::operato
char___std::basic_ostream_char_std::char_traits_char____charconst
0x004016bf      488d154a7c08.   lea rdx, str.HINT            ; 0x489310 ; "\n------------HINT--------
```

La instrucción que provoca que se ejecute una parte u otra del código, se encuentra en la posición 0x004015ec:

0x004015ea **test al, al**
0x004015ec **je 0x401677**

Si hacemos que en radare2 se muestre en pseudocódigo (e asm.pseudo = true) podemos verlo más claro:

```
0x004015ea      84c0            var = al & al
0x004015ec      0f8485000000    if (!var) goto 0x401677
```

Por tanto, nos interesa que esa condición sea "if (var) goto 0x401677", en lugar de que la condición se ejecute con un "NOT var", para que no se cumpla y no vaya a la parte del código donde da el error y siga la ejecución hacia la parte del código que nos da el flag.

Para ello editamos el código ensamblador de esa línea con la instrucción **jne**, que tiene el comportamiento contrario a la instrucción **je**. Con el comando "s" nos situamos en esa línea y con el comando "wa" escribimos en esa línea código ensamblador.

```
[0x00401677]> s 0x004015ec
[0x004015ec]> wa jne 0x401677
Written 6 byte(s) (jne 0x401677) = wx 0f8585000000
[0x004015ec]> pdb
```

Con pdb vemos que se ha efectuado el cambio:

```
  0x004015ea      84c0            test al, al
< 0x004015ec      0f8585000000    jne 0x401677
```

Salimos de radare2 pulsando "q" y ejecutamos de nuevo el archivo modificado, obteniendo el md5 del flag.

```
wine episodio1.exe
Congratulation!!, Stealing Money $$$...
---------------------------
Stolen: 1.000.000.000 $
---------------------------
Flag: e30f35ad8d9cb6efc0778539a669fa85
.........................................
Pulse cualquier tecla para continuar…
```

Por lo tanto el flag es **UAM{e30f35ad8d9cb6efc0778539a669fa85}**

<u>Solución que nos da un flag erróneo:</u>

En lugar de modificar la instrucción "je" por "jne", probamos editando el valor del string "01/23/89" y poner la fecha actual del sistema. Podemos usar un editor hexadecimal o el propio radare2 en modo visual "V".

Pulsando "c" se nos activa un cursor para poder editar, con "Tab" podemos movernos de columna y con "i" entramos en modo "insertar valores".

```
0x00489210  0x00000006 0x0000000a 0x0000000f 0x00000015 |................|
0x00489220  0x00000006 0x0000000a 0x0000000f 0x00000015 |................|
0x00489230  0x00000006 0x0000000a 0x0000000f 0x00000015 |................|
0x00489240  0x312f3530 0x38312f39 0x00000000 0x00000000 |05/19/18........|
0x00489250  0x6e6f430a 0x74617267 0x74616c75 0x216e6f69 |.Congratulation!|
0x00489260  0x53202c21 0x6c616574 0x20676e69 0x656e6f4d |!, Stealing Mone|
0x00489270  0x24242079 0x2e2e2e24 0x2d2d0a00 0x2d2d2d2d |y $$$....------|
0x00489280  0x2d2d2d2d 0x2d2d2d2d 0x2d2d2d2d 0x2d2d2d2d |----------------|
0x00489290  0x2d2d2d2d 0x0a002d2d 0x6c6f7453 0x203a6e65 |------..Stolen: |
0x004892a0  0x30302e31 0x30302e30 0x30302e30 0x20242030 |1.000.000.000 $ |
0x004892b0  0x6c460a00 0x203a6761 0x00000000 0x00000000 |..Flag: ........|
```

Una vez editado el archivo, volvemos a ejecutarlo y nos da un flag, pero no es el correcto.

```
wine episodio1.exe
Congratulation!!, Stealing Money $$$...
--------------------------
Stolen: 1.000.000.000 $
--------------------------
Flag: 0bec9e361fcad73efda45750efc2ddcf
.......................................
Pulse cualquier tecla para continuar...
```

Rafa Martos
@elbuenodefali