

Deviation - 300 pts - Reversing

La descripción del reto nos dice lo siguiente:

Nos dijeron que si corremos este programa con el username correcto nos imprime la bandera, puedes checarlo? **IMPORTANTE:** La bandera empieza con: `hackdef{`

El primer detalle con este reto es que debía hacerse el reversing con ayuda del programa **DnSpy**.

Vemos que desde el **main** se crea un objeto **Flag** al que le pasa como parametro el nombre del usuario que ejecuta el programa y después llama a la función **print** de ese objeto.

```
5 namespace jax
6 {
7     // Token: 0x02000003 RID: 3
8     internal class Program
9     {
10         // Token: 0x06000005 RID: 5 RVA: 0x000021C8 File Offset: 0x00003C8
11         private static void Main(string[] args)
12         {
13             Thread.CurrentThread.CurrentUICulture = new CultureInfo("en-us");
14             string userName = Environment.UserName;
15             Console.WriteLine("Bienvenido {0}. \nPresiona la combinacion correcta para acceder...", userName);
16             if (Console.ReadKey().Key == ConsoleKey.F5 && Console.ReadKey().Key == ConsoleKey.F3 && Console.ReadKey().Key == ConsoleKey.F6)
17             {
18                 Console.WriteLine(new Flag(userName).print());
19             }
20             else
21             {
22                 Console.WriteLine("WOP WOP WOP!!!! Nooooooooooooooooo!");
23             }
24             Console.ReadKey();
25         }
26     }
27 }
```

Dentro del constructor vemos que asigna una flag falsa a una variable miembro **s** pero antes toma el nombre de usuario, lo convierte a un arreglo de char, toma la 8va posición e intenta leer un archivo con ese nombre. Lo interesante aquí es que encierra eso en un **try catch** en donde, si se presenta una **Exception**, llama a una función **l** dando como argumentos la excepción y el nombre del usuario.

```

1  // jax.Flag
2  // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
3  public Flag(string _f)
4  {
5      try
6      {
7          File.ReadAllText(_f.ToCharArray()[8].ToString());
8          this.s = "hackdf{NOP_ESTO_NO_ES_LO_QUE_BUSCAS!}";
9      }
10     catch (Exception e)
11     {
12         this.l(e, _f);
13     }
14 }
15

```

Revisando la función **print** (que se llama desde el **main**) observamos que lo unico que hace es el return de la variable **s**. Por lo que ahora sabemos que la flag se guardará ahí

```

1  // jax.Flag
2  // Token: 0x06000003 RID: 3 RVA: 0x0000219E File Offset: 0x0000039E
3  public string print()
4  {
5      return this.s;
6  }
7

```

Lo cual nos lleva a que es necesario llegar a la función **l** que debe ser la encargada de generar la flag, ya que sino se tendría una flag falsa.

Dentro de la función **l** podemos ver que tipo de excepción es la que necesitamos: **IndexOutOfRangeException** (Así que debe ser por intentar acceder a la posición 8 del nombre, por lo tanto el nombre debe ser de cuando mas **8 caracteres**). También podemos ver que para construir la flag se apoya de el mensaje que viene en la **Exception**, un arreglo de enteros y el resultado de una función **f** a la que se le pasa el nombre de usuario como argumento. Haciendo el análisis de la función **f** concluimos que solo esta agregando x minúsculas a la derecha del nombre de usuario hasta acompletar 8 caracteres.

```

1  // jax.Flag
2  // Token: 0x06000004 RID: 4 RVA: 0x000021A6 File Offset: 0x000003A6
3  private string f(string inp)
4  {
5      if ((inp.Length & 8) <= 0)
6      {
7          return this.f(inp + "x");
8      }
9      return inp;
10 }
11

```

Siguiendo con la construcción de la flag, esta se va armando letra por letra haciendo uso de xor y de una suma. En otras palabras, se cumple que

$$flag[i] = msj[i] \oplus (nums[i] + txt[i \% 8])$$

```

1  // jax.Flag
2  // Token: 0x06000002 RID: 2 RVA: 0x000020A4 File Offset: 0x000002A4
3  private void l(Exception e, string _f)
4  {
5      string message = e.Message;
6      this.s = "";
7      Console.WriteLine(this.s);
8      if (e.GetType() == typeof(IndexOutOfRangeException))
9      {
10         string text = this.f(_f);
11         long num = 137438953472L;
12         int[] array = new int[]
13         { -47, -86, -107, -101, -83, -41, -80, -82, 13, 9, -22, -82, -68, -81, -91, -49, -80,
14           -80, -71, -87, -30, 0, -36, -97, -12, -61, -101, -67, -94, -75, -26, -15 };
15         char[] array2 = message.ToCharArray();
16         char[] array3 = text.ToCharArray();
17         while (((Long)this.s.Length & num) == 0L)
18         {
19             byte b = Convert.ToByte(array2[this.s.Length]);
20             b ^= (byte)((int)Convert.ToByte(array3[this.s.Length % 8]) + array[this.s.Length]);
21             this.s += Convert.ToChar(b).ToString();
22             num >>= 1;
23         }
24         return;
25     }
26     Console.WriteLine("Nop, esa no es la excepcion correcta");
27 }

```

La clave para resolver el reto fue notar que si conocemos los primeros 8 caracteres de la flag (que sí los conocemos dada la nota que dice que la flag comienza con `hackdef{`, que justo son 8 caracteres), entonces podemos conocer los 8 primeros caracteres de `txt` (que de hecho es la longitud de `txt`), esto es porque para toda $0 \leq i < 8$ se cumple que

$$txt[i] = (flag[i] \oplus msj[i]) - nums[i]$$

Ahora solo falta ver cual es el mensaje que trae la **Exception**

```
4 namespace jax
5 {
6     // Token: 0x02000002 RID: 2
7     internal class Flag
8     {
9         // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00002050
10        public Flag(string _f)
11        {
12            try
13            {
14                File.ReadAllText(_f.ToCharArray()[8].ToString());
15                this.s = "hackdf{NOP_ESTO_NO_ES_LO_QUE_BUSCAS!}";
16            }
17            catch (Exception e)
18            {
19                this.l(e, _f);
20            }
21        }
22    }
23    // Token: 0x06000002 RID: 2 RVA: 0x000020A4 File Offset: 0x000020A4
24    private void l(Exception e, string _f)
```

Nombre	Valor
\$exception	{System.IndexOutOfRangeException: Index was outside the bounds of th...
this	{jax.Flag}
_f	"ESCOM"
e	{System.IndexOutOfRangeException: Index was outside the bounds of th...

Incluso podemos confirmar que se estan agregando x al final del nombre de usuario.

```
24 private void l(Exception e, string _f)
25 {
26     string message = e.Message;
27     this.s = "";
28     Console.WriteLine(this.s);
29     if (e.GetType() == typeof(IndexOutOfRangeException))
30     {
31         string text = this.f(_f);
32         long num = 137438953472L;
```

Nombre	Valor
\$exception	{System.IndexOutOfRangeException: Index wa
jax.Flag.f devuelto	"ESCOMxxx"
this	{jax.Flag}
e	{System.IndexOutOfRangeException: Index wa
_f	"ESCOM"
message	"Index was outside the bounds of the array."
text	"ESCOMxxx"
num	0-0000000000000000

Aqui el script usado para generar el nombre de usuario correcto:

```
#!/usr/bin/python
username = "ESCOM"
message = "Index was outside the bounds of the array."
```

```


text = username.ljust(8, "x")
num = 137438953472

array = [-47,-86,-107,-101,-83,-41,-80,-82,13,9,-22,-82,-68,-81,-91,-49,-80,-80,-7
array2 = [ord(x) for x in message]
array3 = [ord(x) for x in text]

flag = "hackdef{"
username = ""
i = 0
while ((i & num) == 0):
    b = ord(flag[i])
    b ^= array2[i]
    b -= array3[i]
    username += chr(b)
    i += 1
    if(i == 8): break
    num >>= 1

print username

```



```

$ ./obt_nombre.py
Personal
$ 

```

Con el cual podemos generar un script para obtener la flag:

```

#!/usr/bin/python
username = "Personal"
message = "Index was outside the bounds of the array."
text = username.ljust(8, "x")
num = 137438953472

array = [-47,-86,-107,-101,-83,-41,-80,-82,13,9,-22,-82,-68,-81,-91,-49,-80,-80,-7
array2 = [ord(x) for x in message]
array3 = [ord(x) for x in text]

flag = ""

```

```
while ((len(flag) & num) == 0):  
    b = array2[len(flag)]  
    b ^= array3[len(flag) % 8] + array[len(flag)]  
    flag += chr(b)  
    num >>= 1  
  
print flag
```

```
$ ./obt_flag.py  
hackdef{.N3T_no_e5_t4N_d1FiC1L!}  
$
```