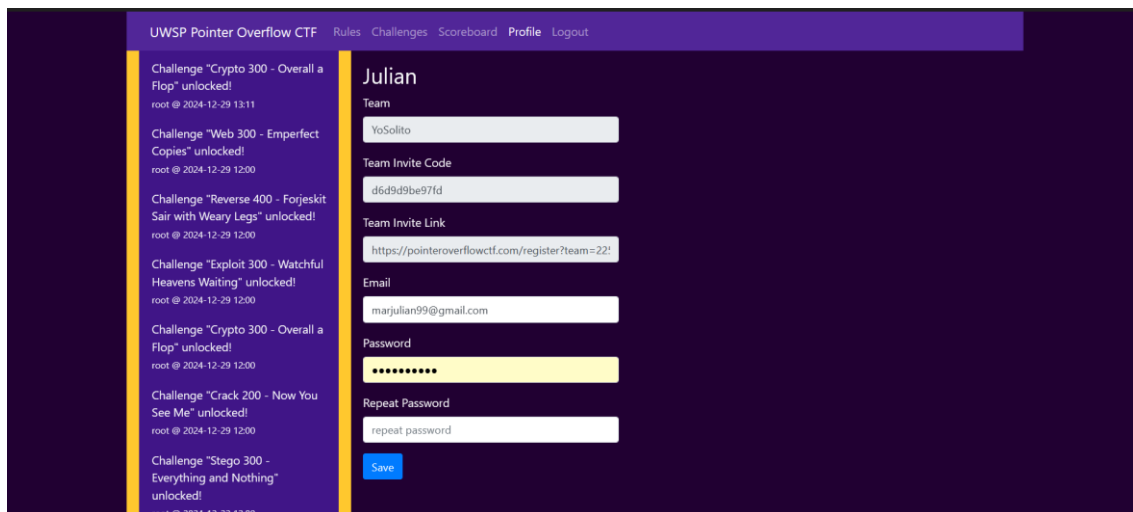


Informe trabajo final Introducción a la Ciberseguridad

Captura del registro del perfil:



The screenshot shows the registration page for the UWSP Pointer Overflow CTF. The page has a dark purple header with navigation links: Rules, Challenges, Scoreboard, Profile, and Logout. The main content area is divided into two columns. The left column lists several challenges that have been unlocked, including "Crypto 300 - Overall a Flop", "Web 300 - Emperfect Copies", "Reverse 400 - Forjeskit Sair with Weary Legs", "Exploit 300 - Watchful Heavens Waiting", "Crypto 300 - Overall a Flop", "Crack 200 - Now You See Me", and "Stego 300 - Everything and Nothing". The right column shows the registration form for a user named Julian. The form includes fields for Team (YoSolito), Team Invite Code (d6d9d9be97fd), Team Invite Link (https://pointeroverflowctf.com/register?team=22), Email (marjulian99@gmail.com), Password (masked with dots), Repeat Password (repeat_password), and a Save button.

Ejercicio 1 Reversing

Nombre del ejercicio: **Reverse 100 - Well Said but Poorly Heard**

Categoría: **Reversing**

Descripción del ejercicio:

So there I was, knee deep in wet newspaper, a puppy under each arm, and a lion hovering over me on the back of a giant dragonfly. Only one thing separating me from certain death at the hands of the President of Murder was the Sphinx and its riddle.

"Ask your riddle, Peptopotamus!" The puppy under my right arm commanded. Its stone face moved, kicking off ancient dust, turning its glowing eyes to us. It spoke its riddle.

"True to false and false to true
What you did before, now undone,
And when you're wrong, reverse your sight.
What am I?"

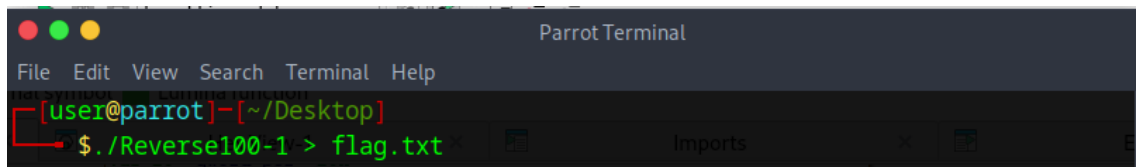
The lion laughed, but he did not know what I knew. He did not know that I created the attached program when I was only 18 months old, and not only did I know the

answer to his riddle - I was its master. Now, I give the program to you, so you too can be prepared if you find yourself in a similar situation.

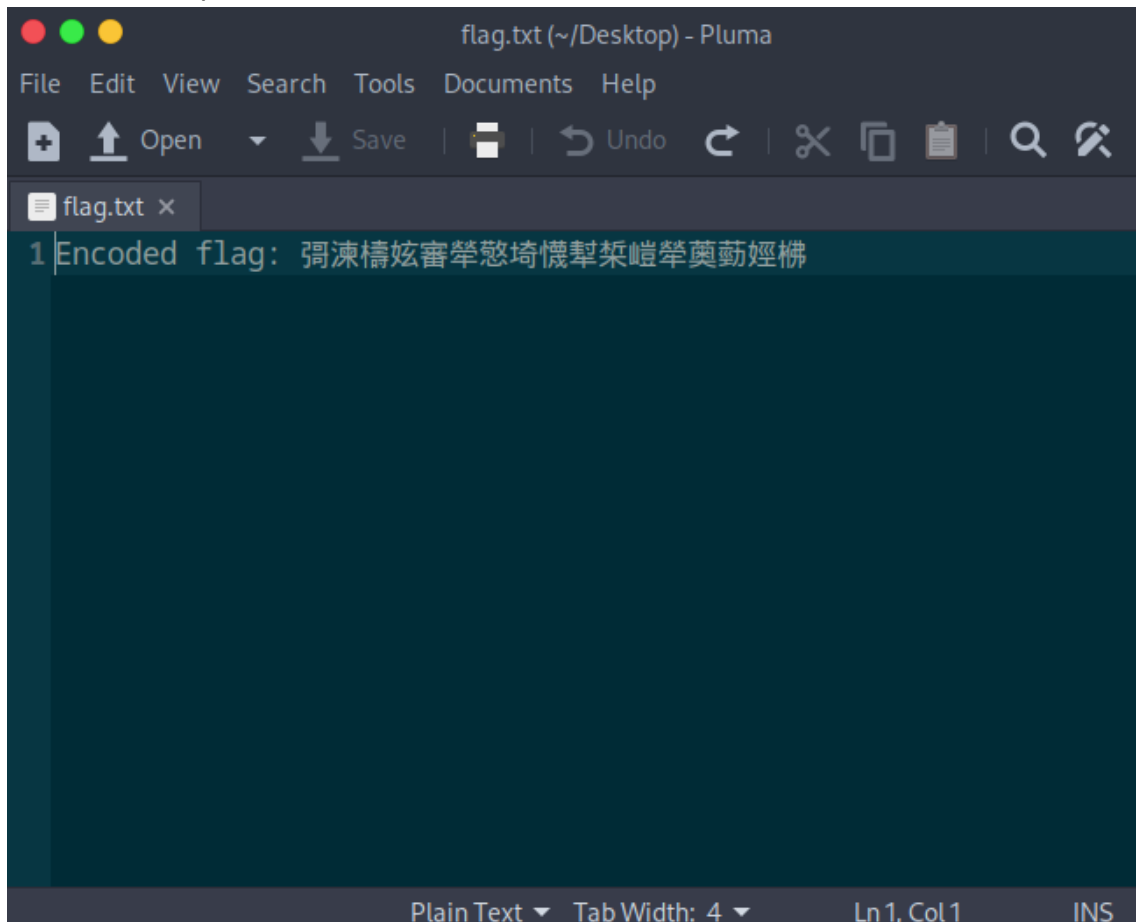


Archivo adjunto:

Lo primero que hice para resolver este ejercicio fue ejecutar el binario guardando el output en un txt

A screenshot of a Parrot Terminal window. The title bar says "Parrot Terminal". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal prompt is "[user@parrot]~[~/Desktop]". The command being executed is "\$./Reverse100-1 > flag.txt".

Esta fue la respuesta obtenida

A screenshot of a text editor window titled "flag.txt (~/Desktop) - Pluma". The menu bar includes "File", "Edit", "View", "Search", "Tools", "Documents", and "Help". The toolbar shows icons for "Open", "Save", "Undo", "Redo", "Cut", "Copy", "Paste", "Find", and "Replace". The text in the editor is "1 Encoded flag: 弼涑禱炫審犖慙埼憊犂桀嶢犖莢蕈經拂". The status bar at the bottom shows "Plain Text", "Tab Width: 4", "Ln 1, Col 1", and "INS".

Dado que la respuesta estaba en chino (literal, no en sentido figurado) probe utilizando el traductor de Google que me dio una traducción errónea y procedí a abrir el binario con IDA obteniendo el siguiente código:

```

; Attributes: bp-based frame

; int __fastcall main(int argc, const char **argv, const char **envp)
public main
main proc near

var_30= qword ptr -30h
var_28= qword ptr -28h
var_20= qword ptr -20h
var_18= qword ptr -18h

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 30h
mov     rax, 77757B6674636F70h
mov     rdx, 31775F6E315F7073h
mov     [rbp+var_30], rax
mov     [rbp+var_28], rdx
mov     rax, 33723368375F336Eh
mov     rdx, 377572375F35315Fh
mov     [rbp+var_20], rax
mov     [rbp+var_18], rdx
mov     dword ptr [rbp+var_18+7], 7D6837h
lea     rax, [rbp+var_30]
mov     rdi, rax
call    obfuscate
lea     rax, [rbp+var_30]
mov     rsi, rax
lea     rax, format          ; "Encoded flag: %s\n"
mov     rdi, rax             ; format
mov     eax, 0
call    _printf
mov     eax, 0
leave
retn
; } // starts at 401155
main endp

_text ends
main+40 (Synchronized with Hex View-1)

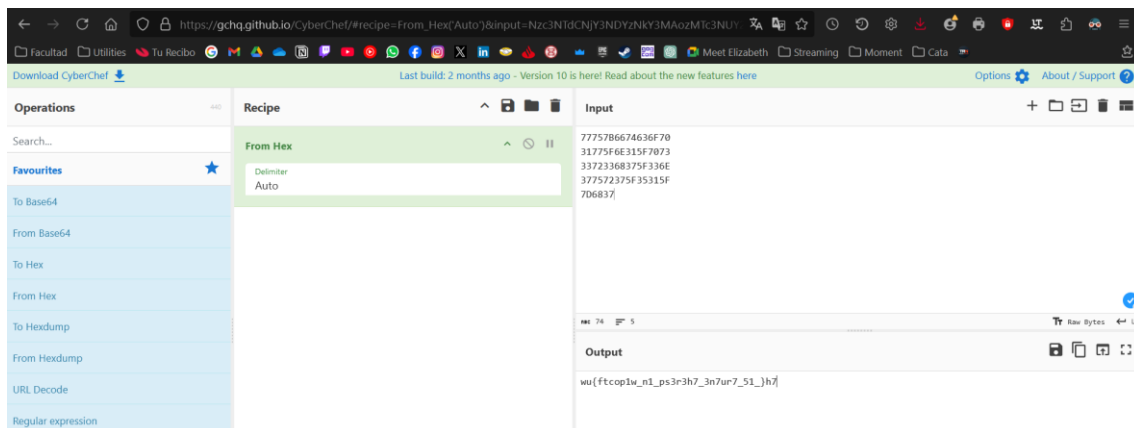
```

```

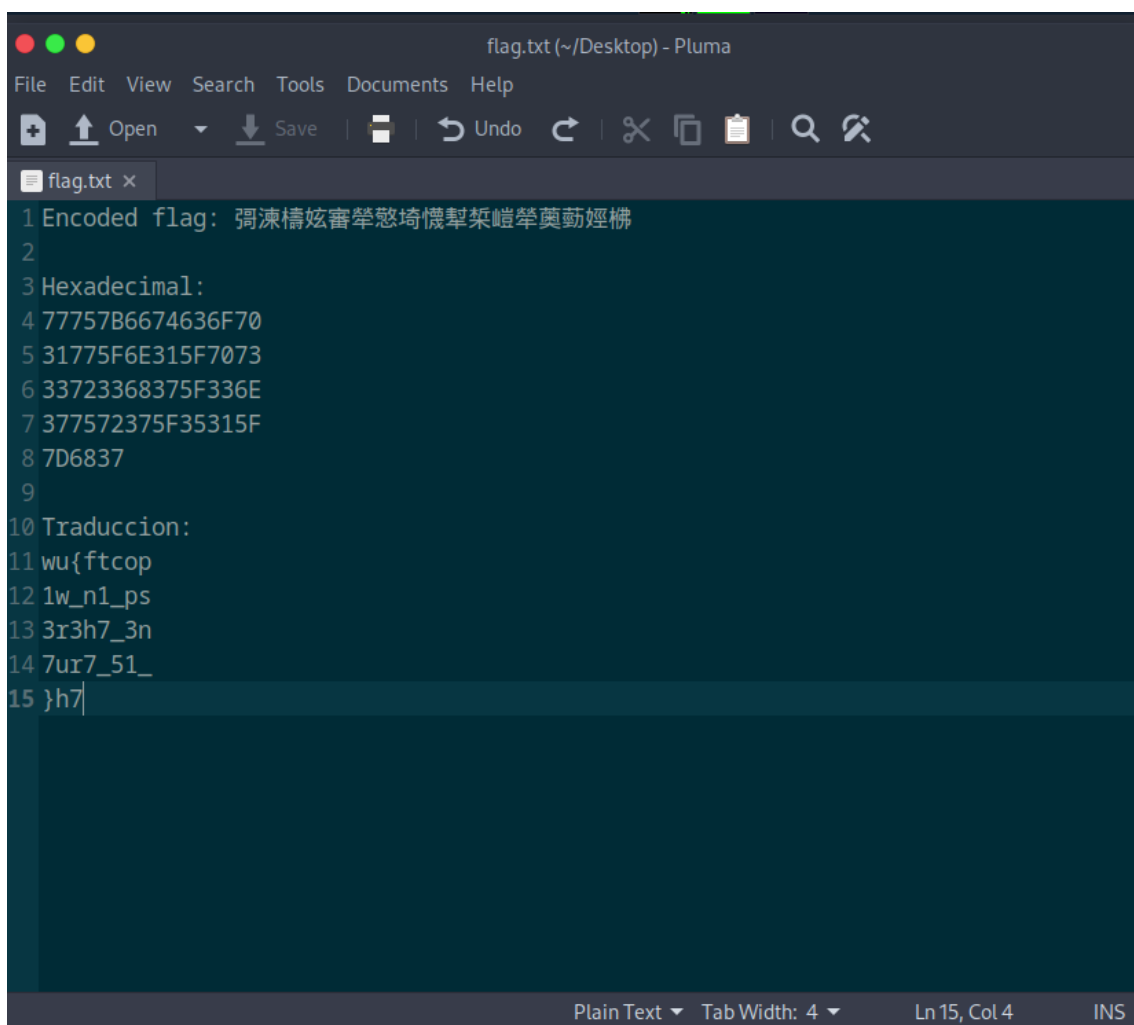
.text:0000000000401126 ; Attributes: bp-based frame
.text:0000000000401126 public obfuscate
.text:0000000000401126 obfuscate proc near ; CODE XREF: main+4Eip
.text:0000000000401126 var_8 = qword ptr -8
.text:0000000000401126 ; __unwind {
.text:0000000000401126 push rbp
.text:0000000000401127 mov rbp, rsp
.text:0000000000401128 mov [rbp+var_8], rdi
.text:0000000000401128 jmp short loc_401146
.text:0000000000401130 ;
.text:0000000000401130 loc_401130: mov rax, [rbp+var_8] ; CODE XREF: obfuscate+29j
.text:0000000000401134 movzx eax, byte ptr [rax]
.text:0000000000401137 not eax
.text:0000000000401139 mov edx, eax
.text:000000000040113B mov rax, [rbp+var_8]
.text:000000000040113F mov [rax], dl
.text:0000000000401141 add [rbp+var_8], 1
.text:0000000000401146 ;
.text:0000000000401146 loc_401146: mov rax, [rbp+var_8] ; CODE XREF: obfuscate+8j
.text:000000000040114A movzx eax, byte ptr [rax]
.text:000000000040114D test al, al
.text:000000000040114F jnz short loc_401130
.text:0000000000401151 nop
.text:0000000000401152 nop
.text:0000000000401153 pop rbp
.text:0000000000401154 retn
.text:0000000000401154 ; } // starts at 401126
.text:0000000000401154 obfuscate endp

```

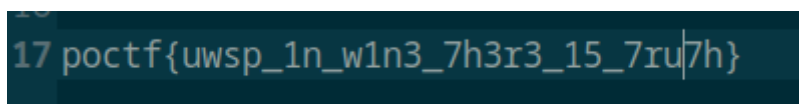
El código lo que hace es invertir los bits de cada byte de datos en la función obfuscate. La flag estaba escrita en el código previo a la traducción en hexadecimal, por lo que la copie directamente del binario el hexadecimal y la traduje en cyberchef



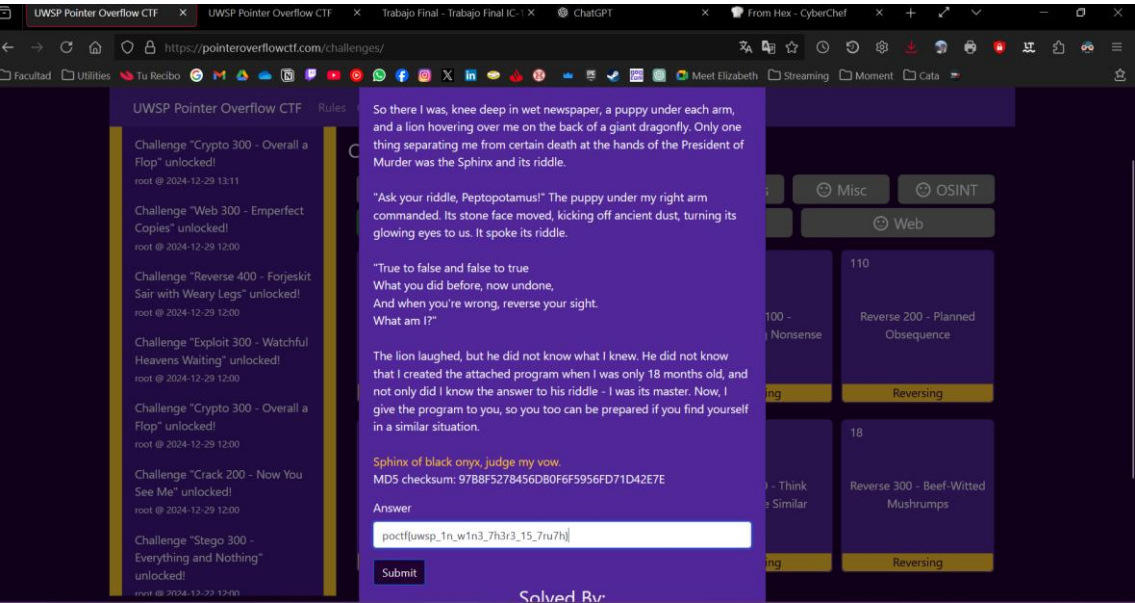
Acá identifique que la flag estaba escrita desordenada y traduje línea por línea quedando de la siguiente manera:

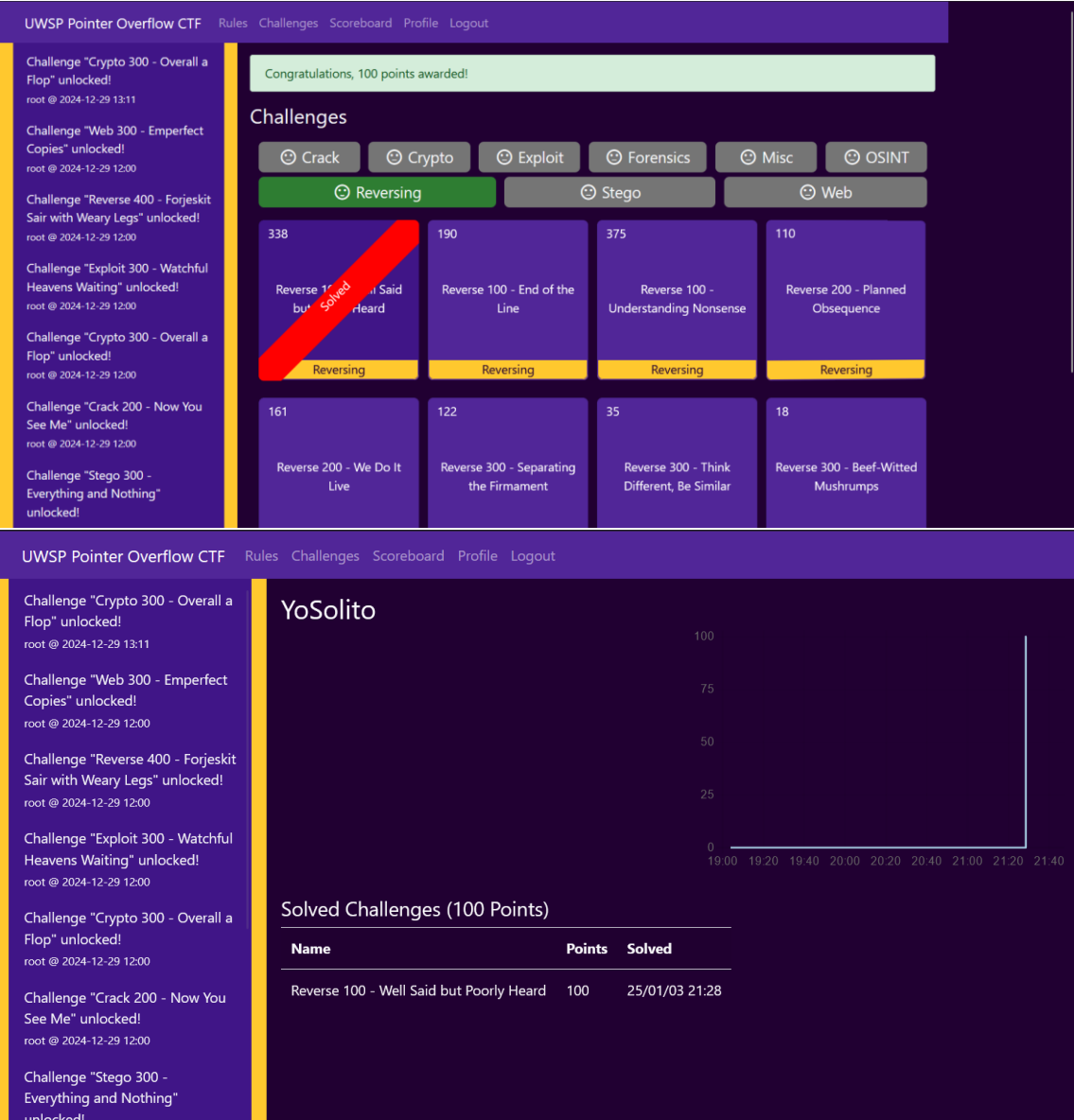


Acá podía verse que la flag esta desordenada. Ordene los datos obtenidos según el formato necesario de la pagina y quedo la siguiente flag final de respuesta:



Capturas evidencia de resolución:





Ejercicio 2 Reversing

Nombre del ejercicio: Reverse 100 - Well Said but Poorly Heard

Categoria: Reversing

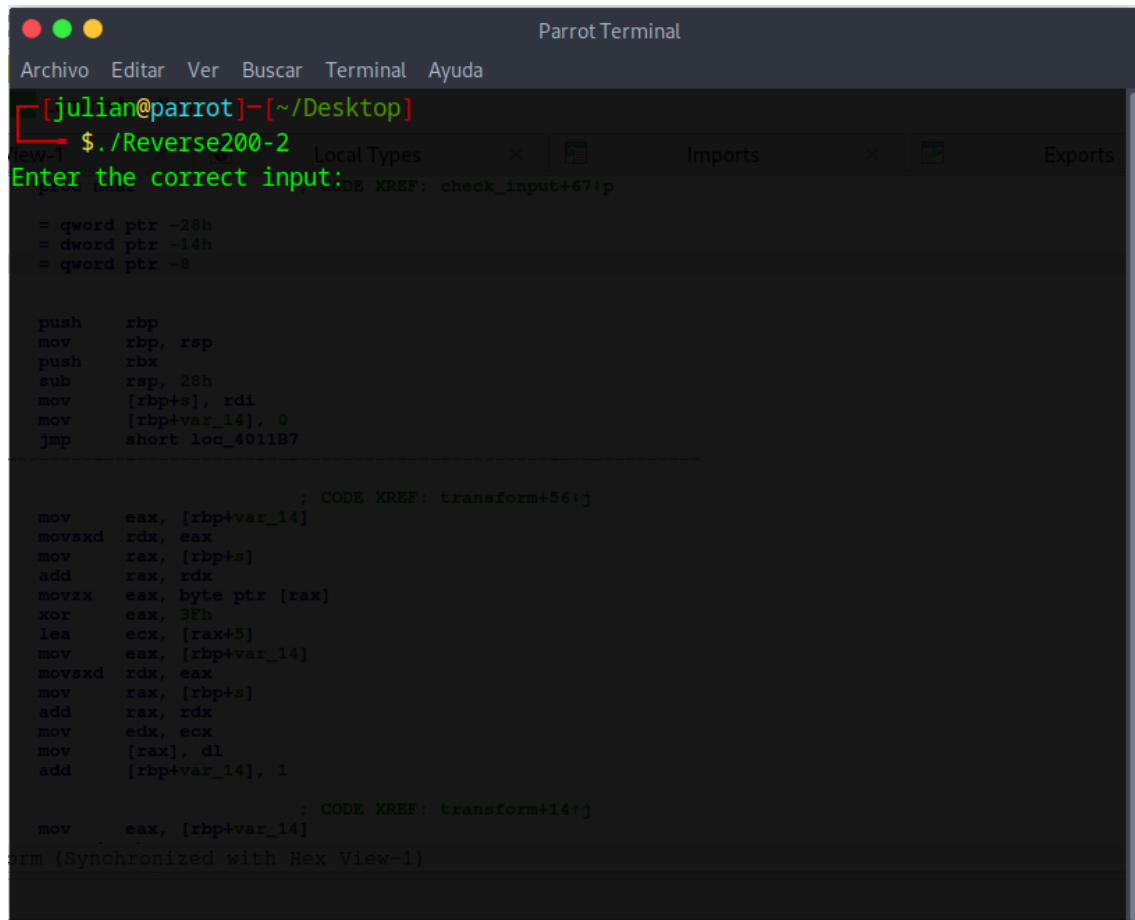
Descripcion del ejercicio: You know what I really hate? Challenges where you need to breakdown a function and then script your own solution. Who has time for that?? Anyway, let's do that! Luckily this should be a pretty quick one because I have got so much TV to catch up on.



Reverse200-2

Archivo adjunto:

El ejecutable lo que hace es pedirnos ingresar un input para poder revelar la flag



```
[julian@parrot]-[~/Desktop]
$ ./Reverse200-2
Enter the correct input:
; CODE XREF: check_input+67+p

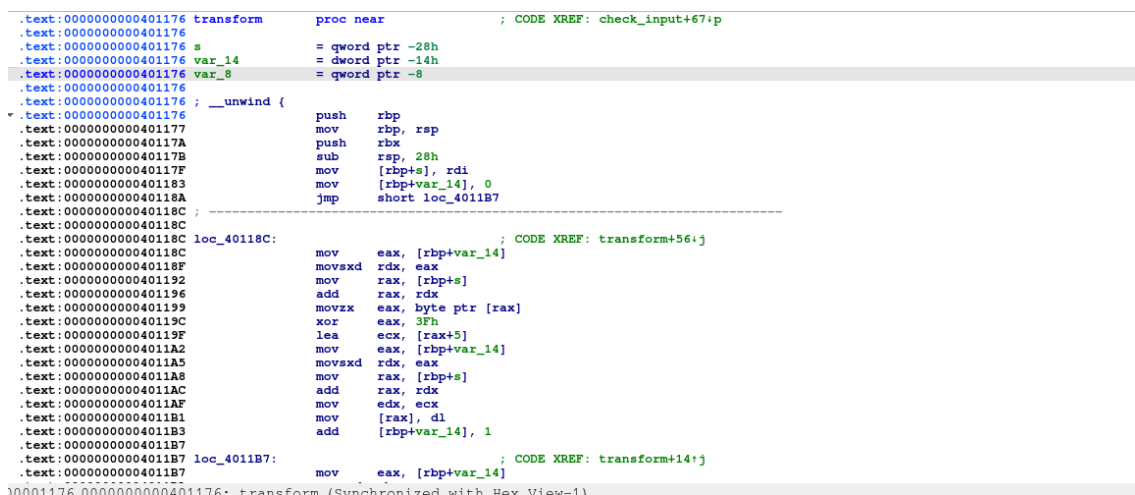
= qword ptr -28h
= dword ptr -14h
= qword ptr -8

push    rbp
mov     rbp, rsp
push    rbx
sub     rsp, 28h
mov     [rbp+s], rdi
mov     [rbp+var_14], 0
jmp     short loc_4011B7

; CODE XREF: transform+56+j
mov     eax, [rbp+var_14]
movsxd  rdx, eax
mov     rax, [rbp+s]
add     rax, rdx
movzx   eax, byte ptr [rax]
xor     eax, 3Fh
lea     ecx, [rax+5]
mov     eax, [rbp+var_14]
movsxd  rdx, eax
mov     rax, [rbp+s]
add     rax, rdx
mov     edx, ecx
mov     [rax], dl
add     [rbp+var_14], 1

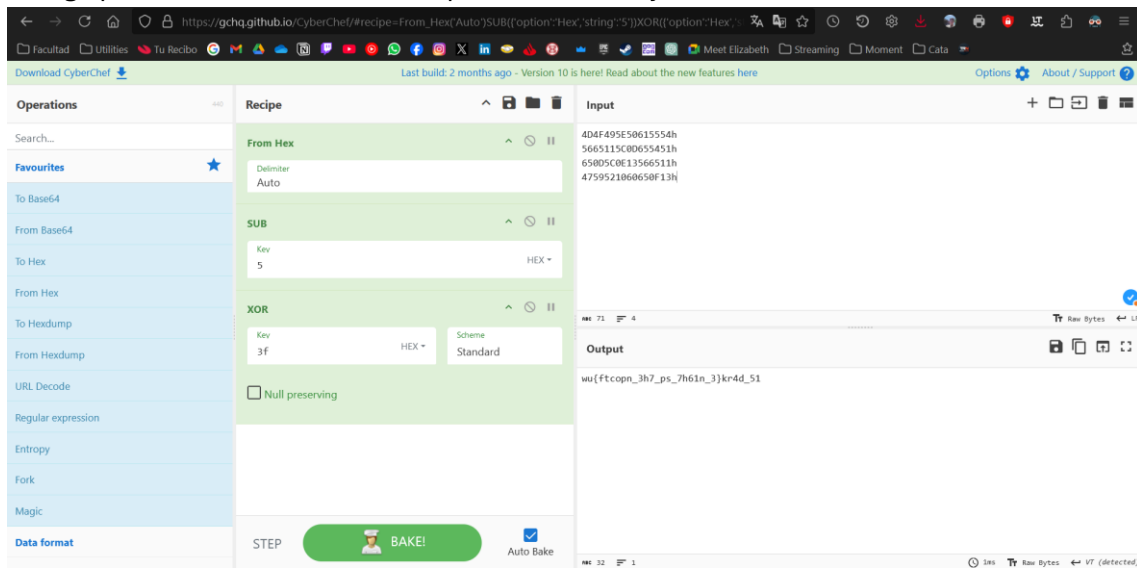
; CODE XREF: transform+14+j
mov     eax, [rbp+var_14]
int3    (Synchronized with Hex View-1)
```

Abri el archivo binario con IDA para buscar cual era el input deseado y descubri que el mismo estaba en la memoria. Lo que hacia el código era a mi input aplicarle una función transform que aplicaba un XOR y le sumaba 5 para luego compararla con el valor que tenia en memoria.

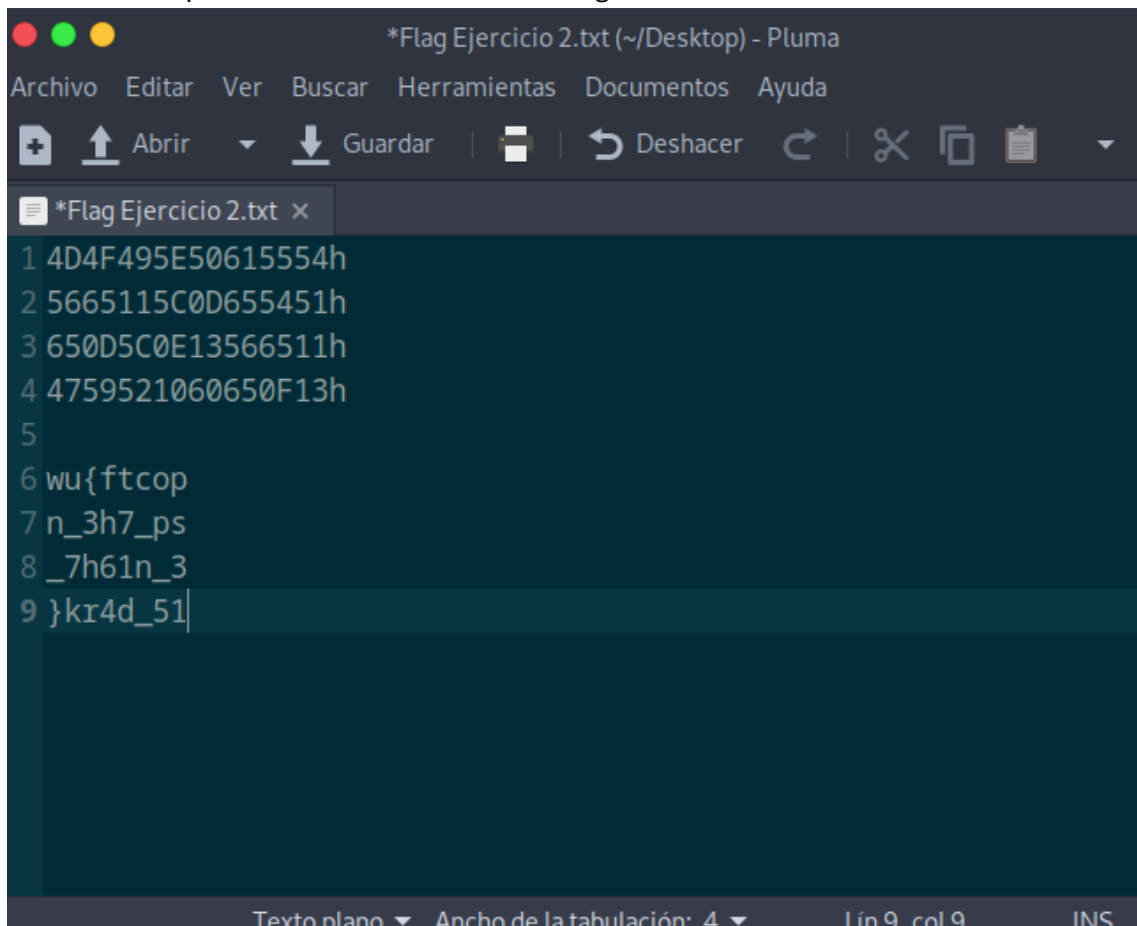


```
.text:0000000000401176 transform      proc near          ; CODE XREF: check_input+67+p
.text:0000000000401176                                     = qword ptr -28h
.text:0000000000401176 s                                     = dword ptr -14h
.text:0000000000401176 var_14                             = qword ptr -8
.text:0000000000401176 var_8
.text:0000000000401176 ; __unwind {
.text:0000000000401176     push    rbp
.text:0000000000401177     mov     rbp, rsp
.text:000000000040117A     push    rbx
.text:000000000040117B     sub     rsp, 28h
.text:000000000040117F     mov     [rbp+s], rdi
.text:0000000000401183     mov     [rbp+var_14], 0
.text:000000000040118A     jmp     short loc_4011B7
.text:000000000040118C ;
.text:000000000040118C loc_40118C:
.text:000000000040118C     mov     eax, [rbp+var_14] ; CODE XREF: transform+56+j
.text:000000000040118F     movsxd  rdx, eax
.text:0000000000401192     mov     rax, [rbp+s]
.text:0000000000401196     add     rax, rdx
.text:0000000000401199     movzx   eax, byte ptr [rax]
.text:000000000040119C     xor     eax, 3Fh
.text:000000000040119F     lea     ecx, [rax+5]
.text:00000000004011A2     mov     eax, [rbp+var_14]
.text:00000000004011A5     movsxd  rdx, eax
.text:00000000004011A8     mov     rax, [rbp+s]
.text:00000000004011AC     add     rax, rdx
.text:00000000004011AF     mov     edx, ecx
.text:00000000004011B1     mov     [rax], dl
.text:00000000004011B3     add     [rbp+var_14], 1
.text:00000000004011B7 loc_4011B7:
.text:00000000004011B7     mov     eax, [rbp+var_14] ; CODE XREF: transform+14+j
10001176 0000000000401176 transform (Synchronized with Hex View-1)
```

Asique decidí aplicar la función inversa al hexadecimal que estaba almacenado en el código para descubrir cual era el input, utilizando cyberchef.



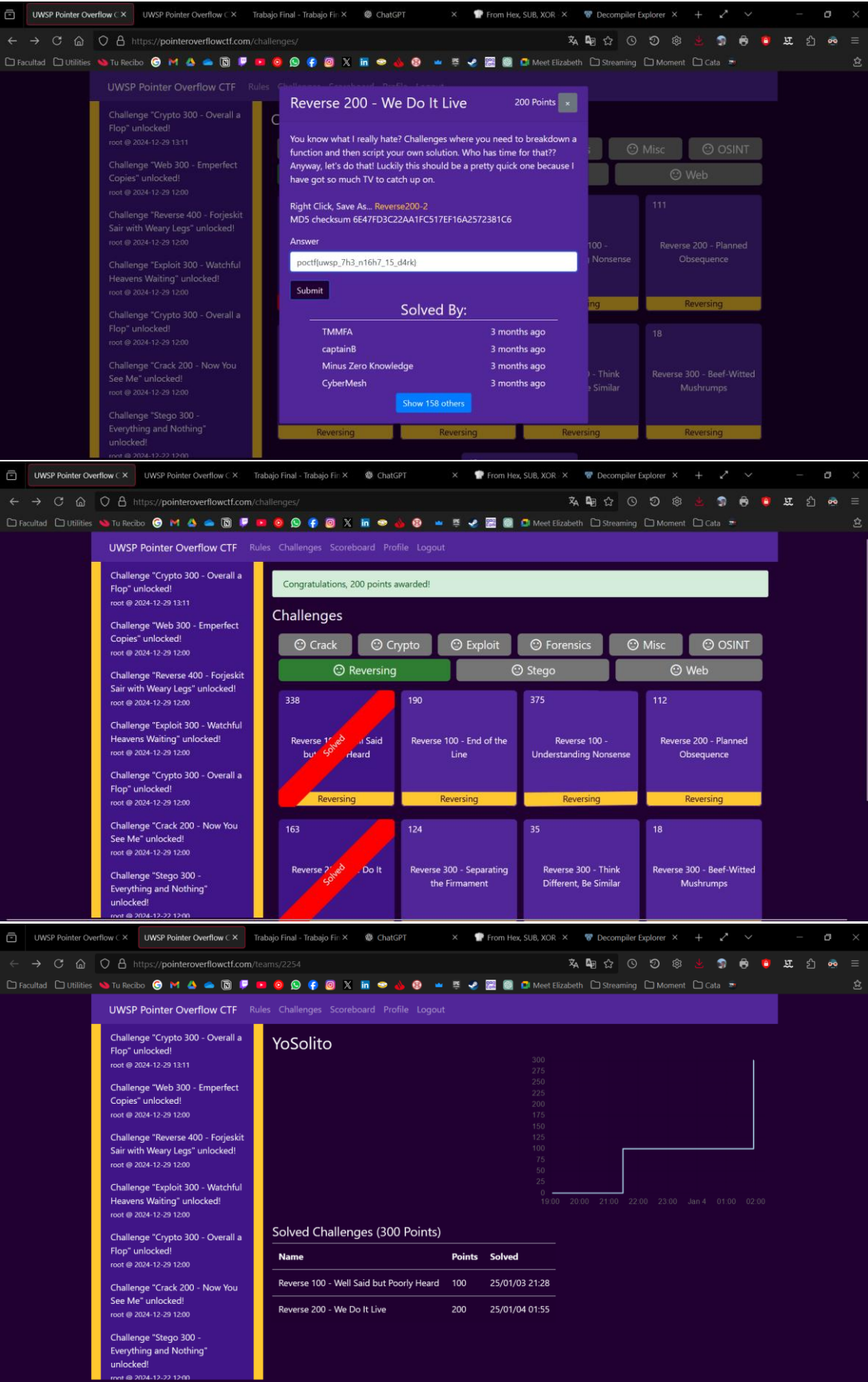
Reescribi los outputs uno por uno en un txt para poder reordenarlos y termine encontrando que el dato almacenado era la flag final



Luego de reordenar los outputs me quedo la siguiente flag final

```
*Flag Ejercicio 2.txt x
1 4D4F495E50615554h
2 5665115C0D655451h
3 650D5C0E13566511h
4 4759521060650F13h
5
6 wu{ftcop
7 n_3h7_ps
8 _7h61n_3
9 }kr4d_51
10
11 poctf{uwsp_7h3_n16h7_15_d4rk}
```

Capturas evidencia de resolución:



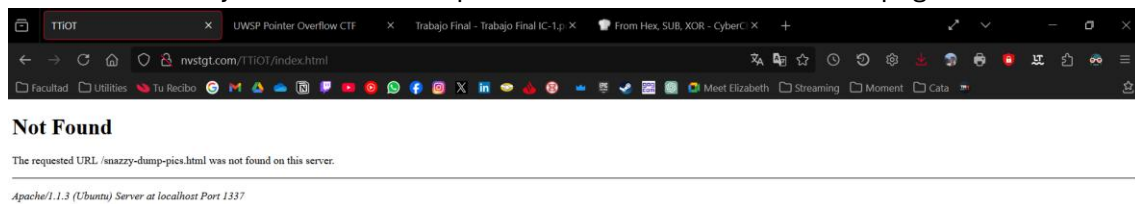
Ejercicio 3 Web

Nombre del ejercicio: **Web 100 - The Way Out is Through**

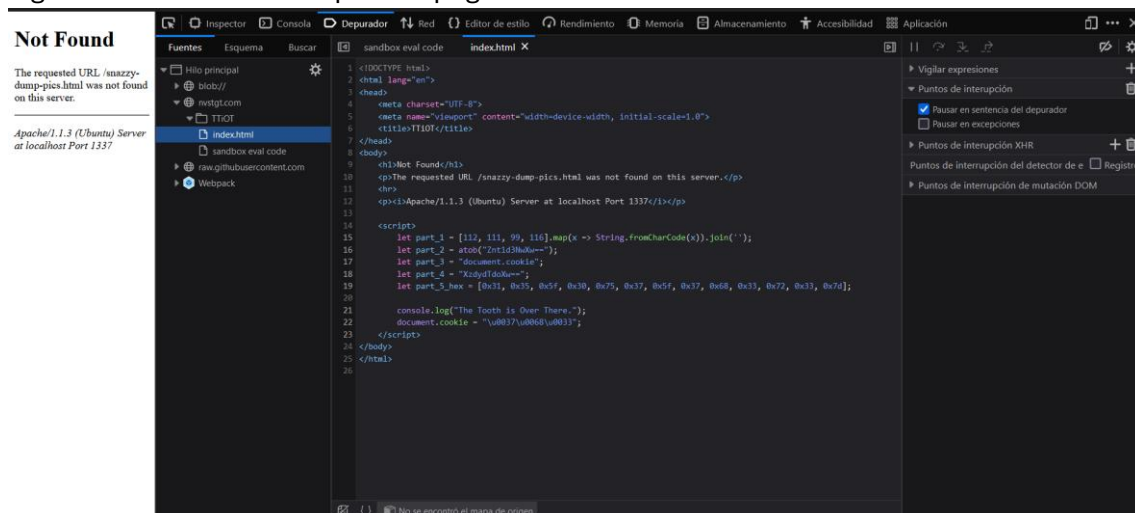
Descripcion: The first Web challenge of the contest and I am so excited to reveal this one! In this challenge you'll run through a simulated web-based cybersecurity training course a la the DoD Cyber Exchange Awareness Challenge. There's just one problem... The cybersecurity training is... Vulnerable??? Oh, the irony! Can YOU handle the HACK OF THE CENTURY??? Head here to find out!

Link adjunto: <http://nvstgt.com/TTiOT/index.html>

Para resolver el ejercicio abri el link provisto encontrándome con una pagina con error 404



Decidi abrir la opcion de inspeccionar elementos y me dirigi al depurador encontrando los siguientes datos en el script de la pagina



Download CyberChef
Last build: 2 months ago - Version 10 is here! Read about the new features here
Options
About / Support

Operations
Search...
Favourites
To Base64
From Base64
To Hex
From Hex
To Hexdump
From Hexdump
URL Decode
Regular expression
Entropy

Recipe
From Base64
Alphabet: A-Za-z0-9+/=
☒ Remove non-alphabet chars
☐ Strict mode

Input
Znt1d3h0Xw==
Output
f{unsp_}

Download CyberChef
Last build: 2 months ago - Version 10 is here! Read about the new features here
Options
About / Support

Operations
unic
Escape Unicode Characters
Unescape Unicode Characters
Normalise Unicode
Unicode Text Format
AES Key Unwrap
AES Key Wrap
Colossus
Enigma
Escape string

Recipe
Unescape Unicode Characters
Prefix: \u

Input
\u0037\u0068\u0033
Output
7h3

Download CyberChef
Last build: 2 months ago - Version 10 is here! Read about the new features here
Options
About / Support

Operations
Search...
Favourites
To Base64
From Base64
To Hex
From Hex
To Hexdump
From Hexdump
URL Decode
Regular expression
Entropy

Recipe
From Base64
Alphabet: A-Za-z0-9+/=
☒ Remove non-alphabet chars
☐ Strict mode

Input
Xzdyd1doXw==
Output
7ru7h}

Download CyberChef
Last build: 2 months ago - Version 10 is here! Read about the new features here
Options
About / Support

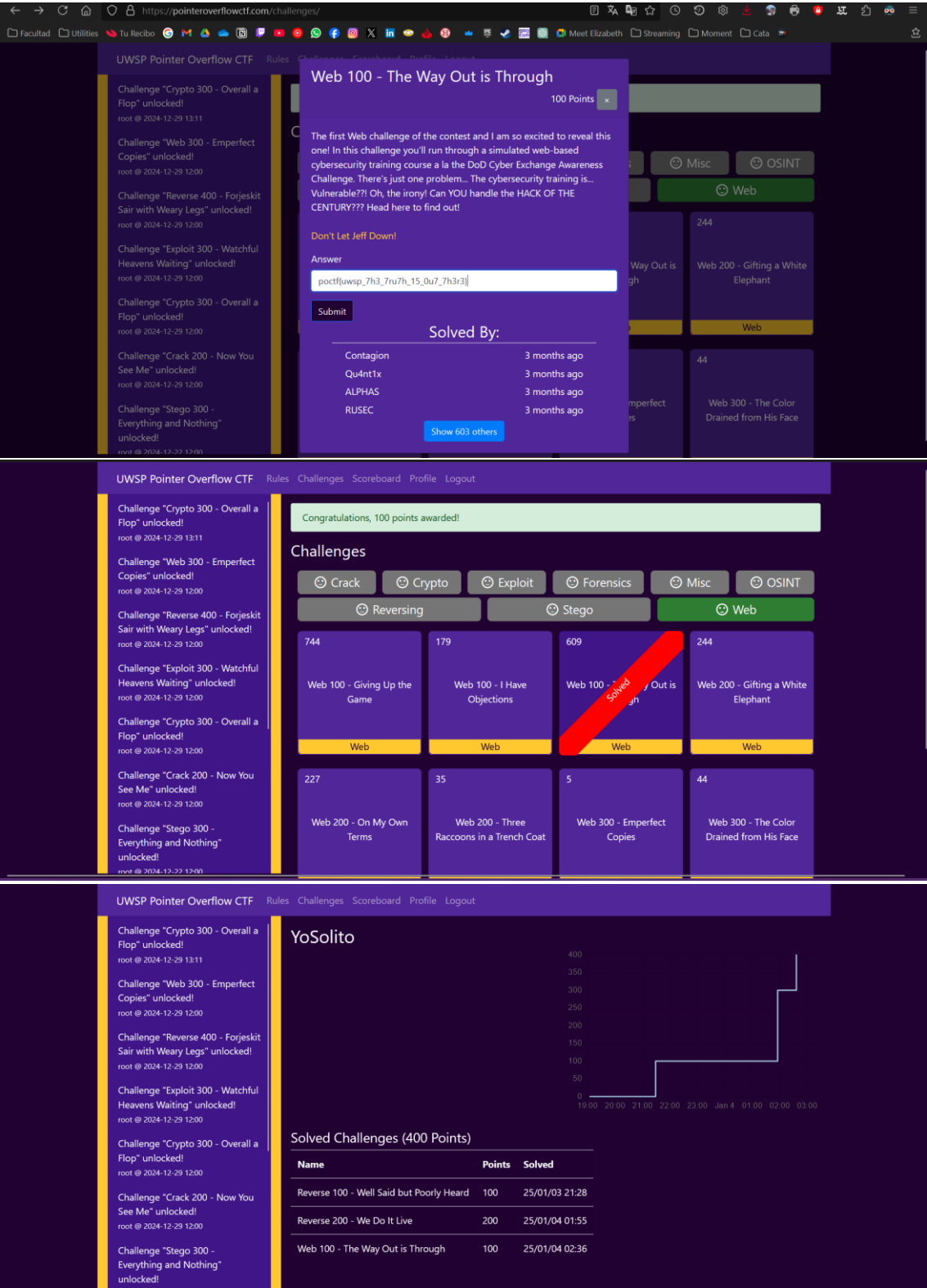
Operations
Search...
Favourites
To Base64
From Base64
To Hex
From Hex
To Hexdump
From Hexdump
URL Decode
Regular expression
Entropy

Recipe
From Hex
Delimiter: Auto

Input
0x31, 0x35, 0x5f, 0x30, 0x75, 0x37, 0x5f, 0x37, 0x68, 0x33, 0x72, 0x33, 0x7d
Output
15_0u7_7h3r3}

Como resultado final la flag fue: p0ctf{unsp_7h3_7ru7h_15_0u7_7h3r3}

Capturas evidencia de resolucion:



Conclusion

A lo largo del desafío de los CTF presentados en este torneo, resulto interesante el hecho de aplicar lo aprendido en la materia en un contexto diferente. Pude resolver los ejercicios de forma satisfactoria y hasta probar algunos de los demás que no adjunte en esta entrega. Me resulto divertido en cierta manera por algunas de las temáticas creativas que se aplicaban e incluso encontrándome con algo tan raro como un texto en chino previo a descriptar la flag. Considero que fue una buena experiencia y que me quedan las ganas de en algún momento participar de un CTF junto con un equipo.