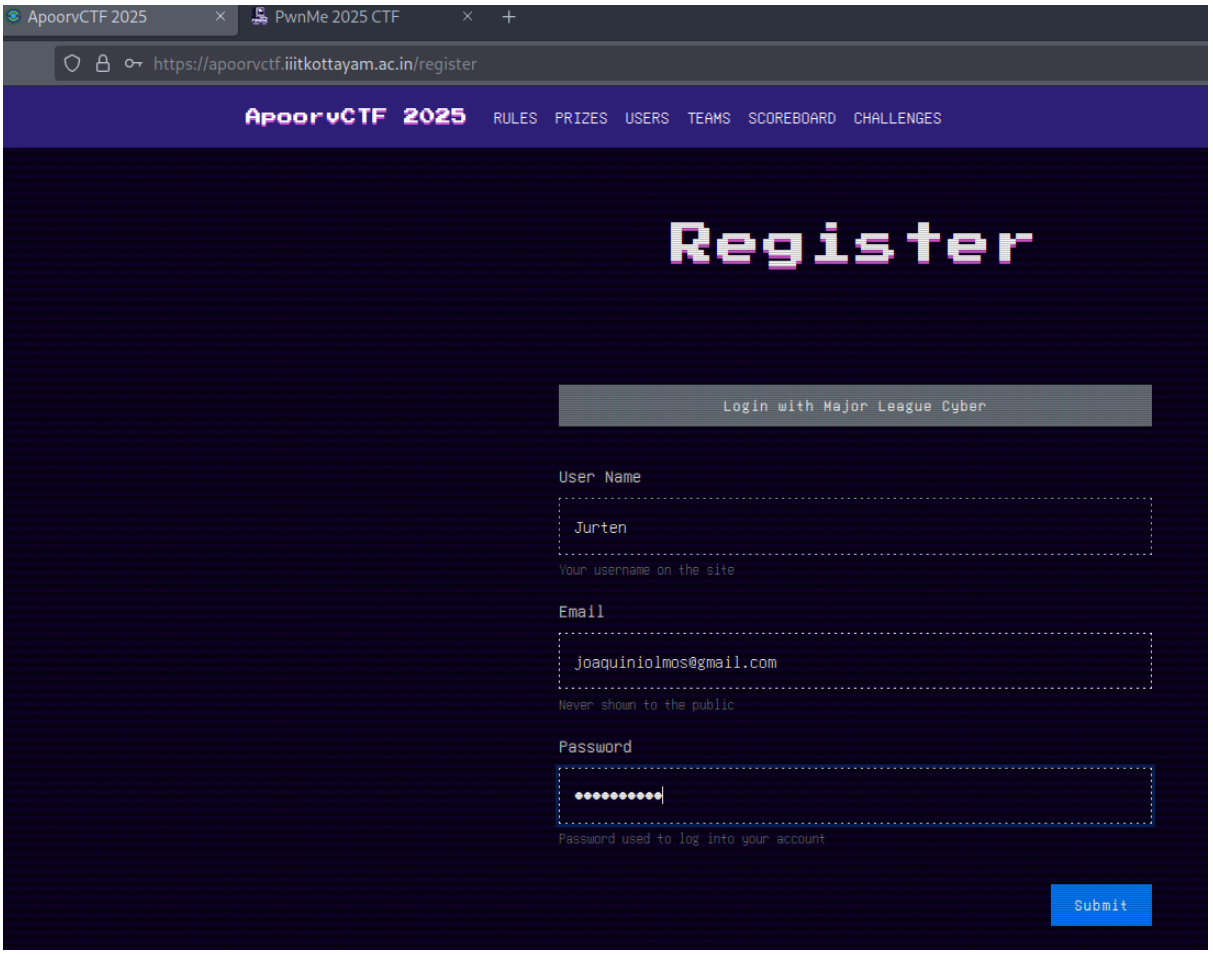
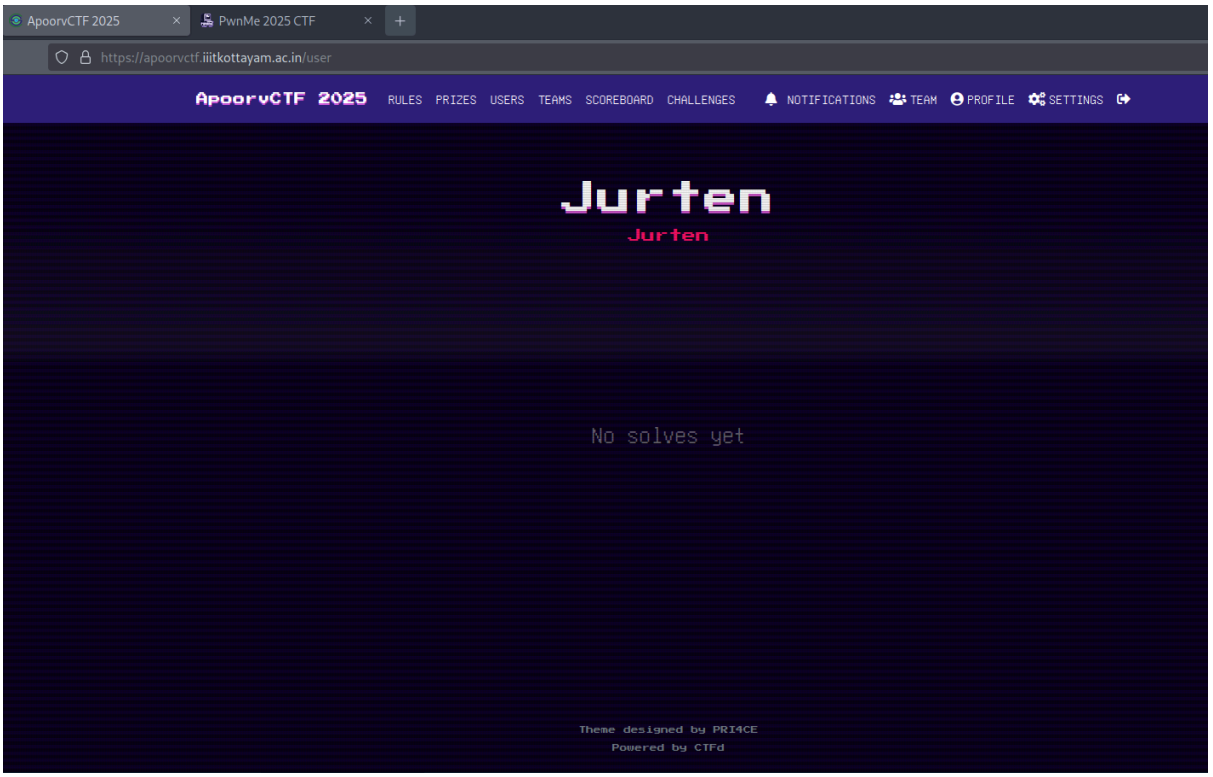
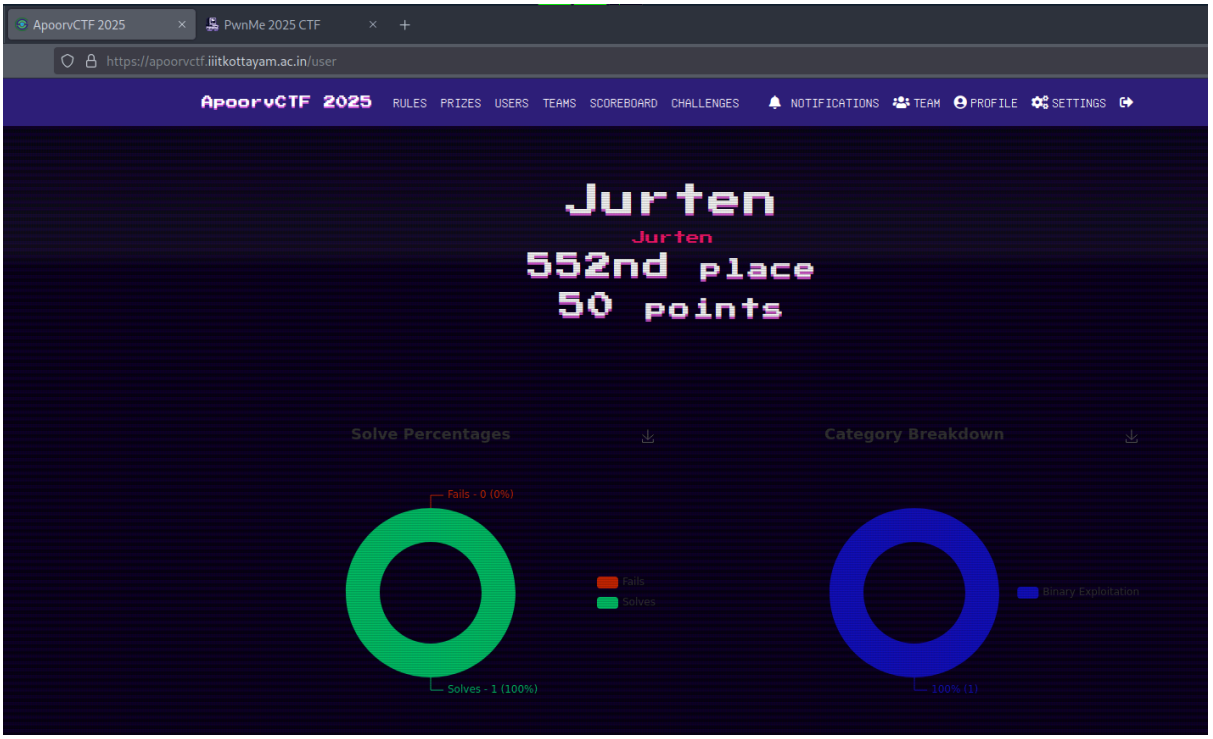
ApoorvCTF:

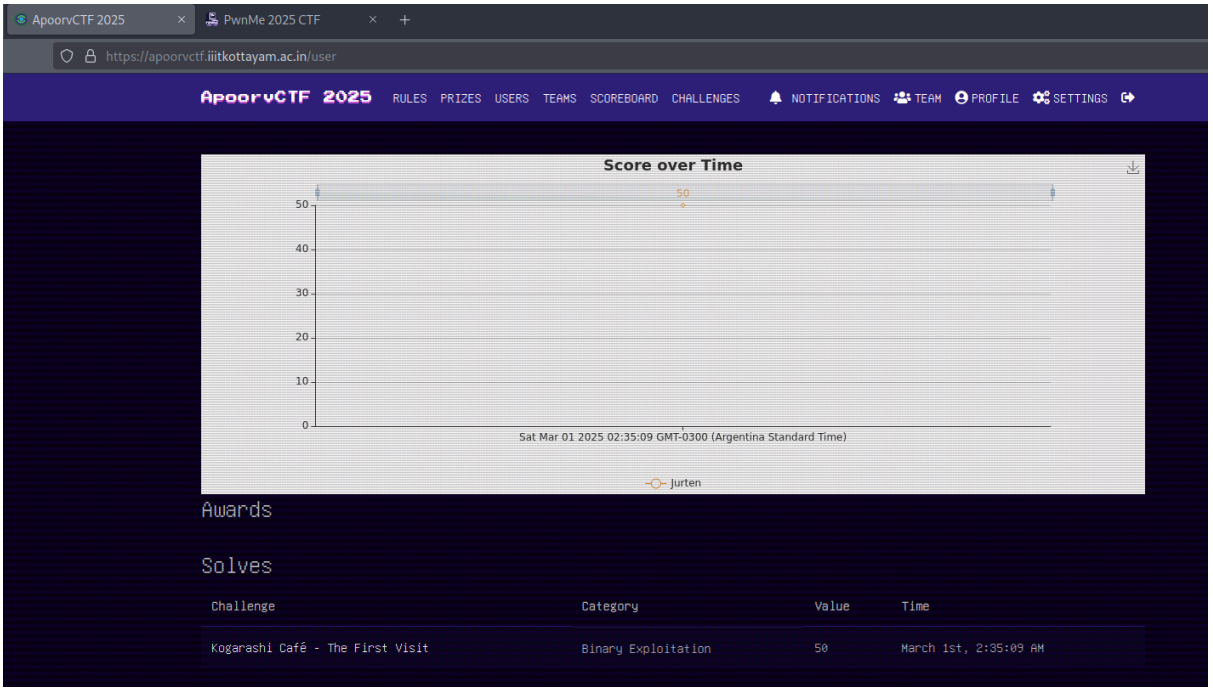

ApoorvCTF 2025
PwnMe 2025 CTF
https://apoorvctf.iiitkottayam.ac.in/register
ApoorvCTF 2025
RULES
PRIZES
USERS
TEAMS
SCOREBOARD
CHALLENGES
Register
Login with Major League Cyber
User Name
Jurten
Your username on the site
Email
joaquiniolmos@gmail.com
Never shown to the public
Password
Password used to log into your account
Submit

The First Visit (PWN):

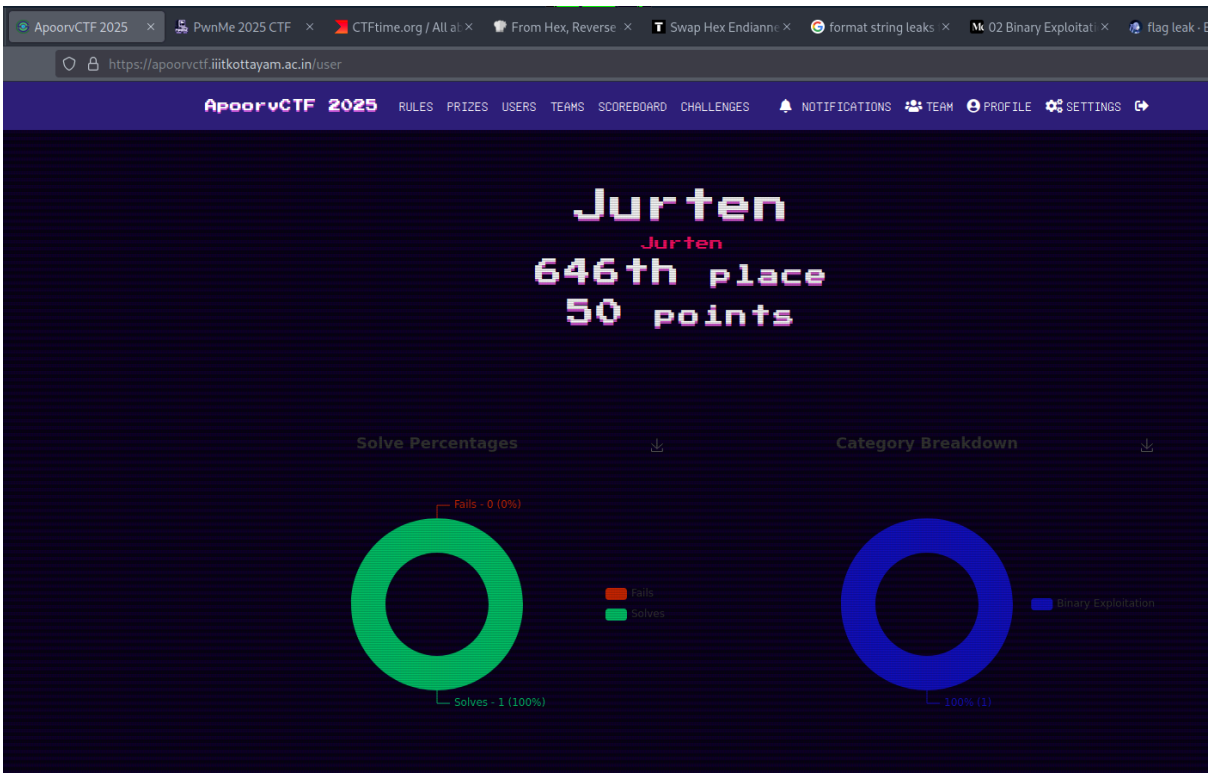


era un ret2win, me rompi la cabeza pensando que era un ret2libc hasta que una funcion rara que se llamaba brew_coffee() y cuando la decompile devolvia la flag… Simplemente cargue el elf para saber donde estaban los symbols, y como es un archivo sin PIE las mismas direcciones que tuviera local iban a estar remotas, asi que era cuestion de poner en la direccion de retorno la direccion de la funcion y tada, la flag salia sola.
flag:apoorvctf{c0ffee_buff3r_sp1ll}



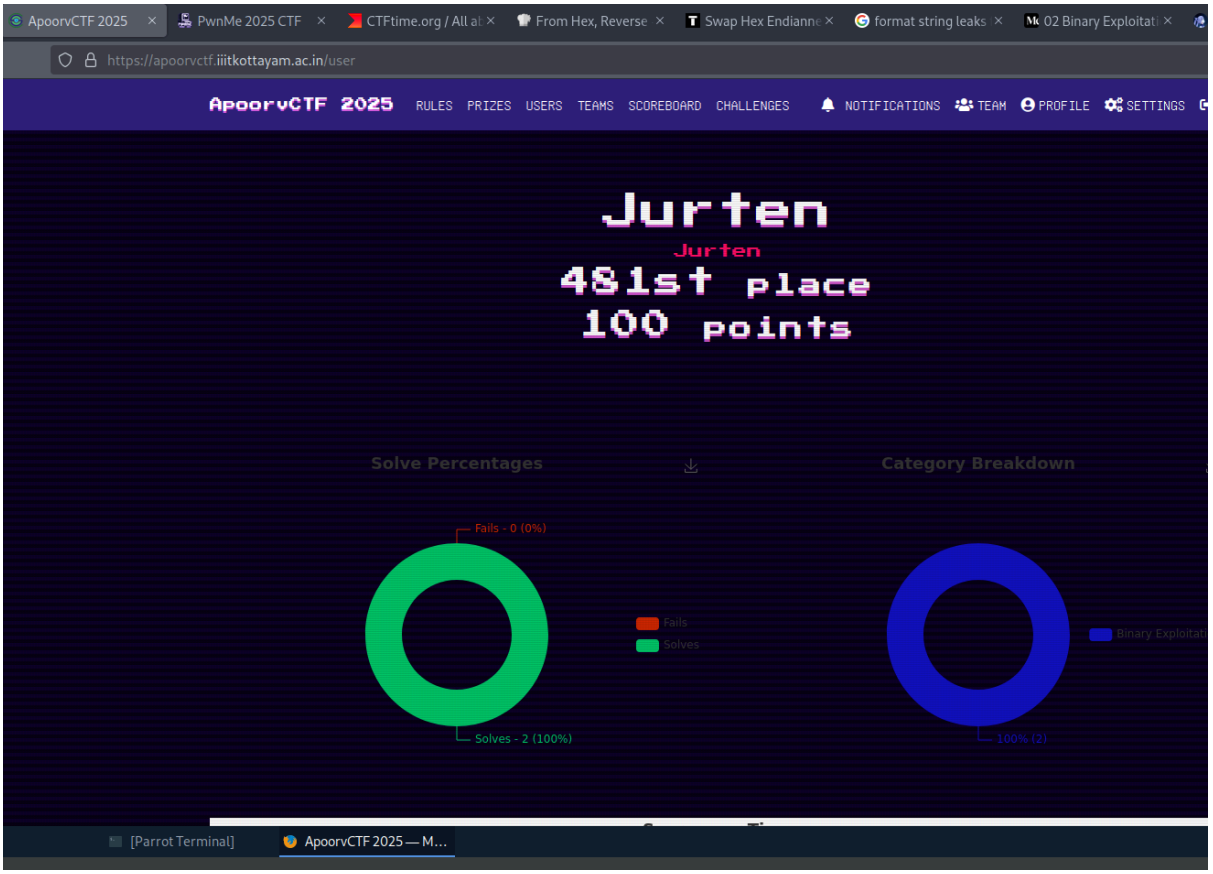


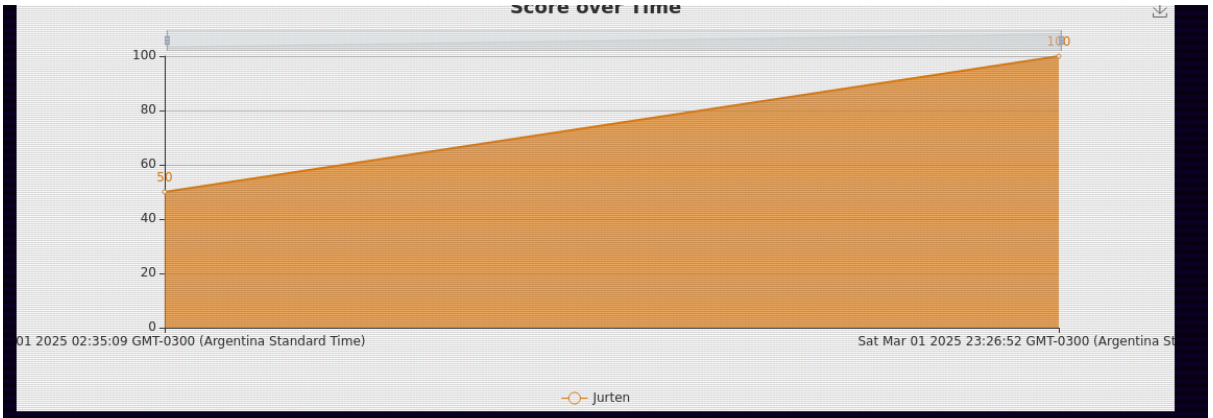
The Secret Blend (PWN):



En este fue facil ver que era un ejercicio de format string porque tenia la flag guardada en un puntero y facilmente le metias diferentes opciones como %x, %p y te leakeaba memoria, fue cosa de concatenar varios %p y darme cuenta que podia sacar la flag con eso y escribir un script que recorra la memoria y teniendo en cuenta el endianness del programa me armara la flag en un formato legible.

flag: apoorvctf{Th3_M3nu_L34ks_M0re_Than_It_Sh0uld}





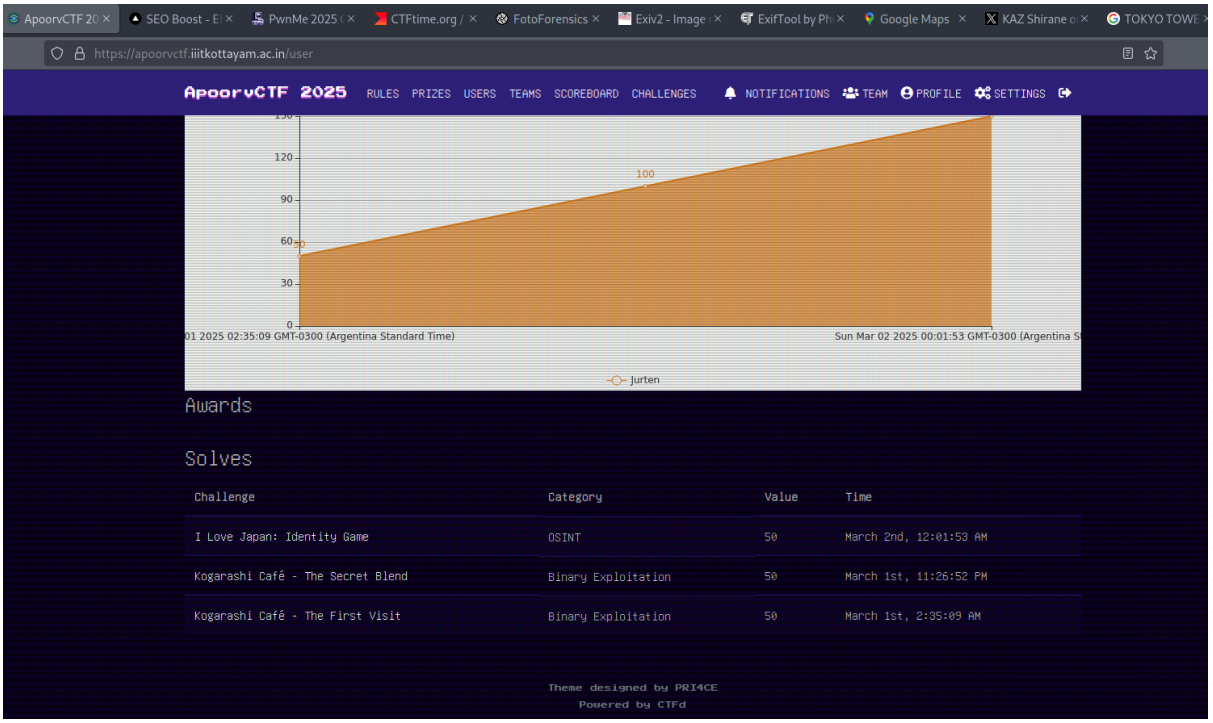
Awards

Solves

Challenge	Category	Value	Time
Kogarashi Café - The Secret Blend	Binary Exploitation	50	March 1st, 11:26:52 PM
Kogarashi Café - The First Visit	Binary Exploitation	50	March 1st, 2:35:09 AM

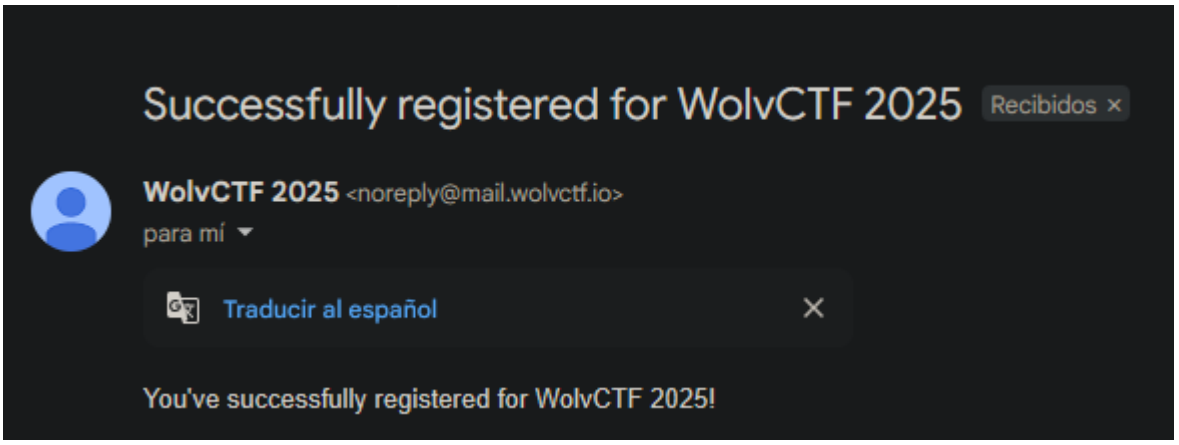
I Love Japan (OSINT):

Aca le erre mucho pensando que tenía que sacar el nombre del fotógrafo y era el de la diseñadora de la Tokyo Tower Top Deck, que una vez que buscabas en google la foto te salia el nick y con eso le pregunte a chatGPT, le deje que busque en internet y enseguida me encontró el nombre.
flag: apoorvctf{masakazu_shirane}



wolvctf:

Me olvidé de sacar foto al registro. Esta es la foto de confirmación de registro



p0wn3d(PWN):

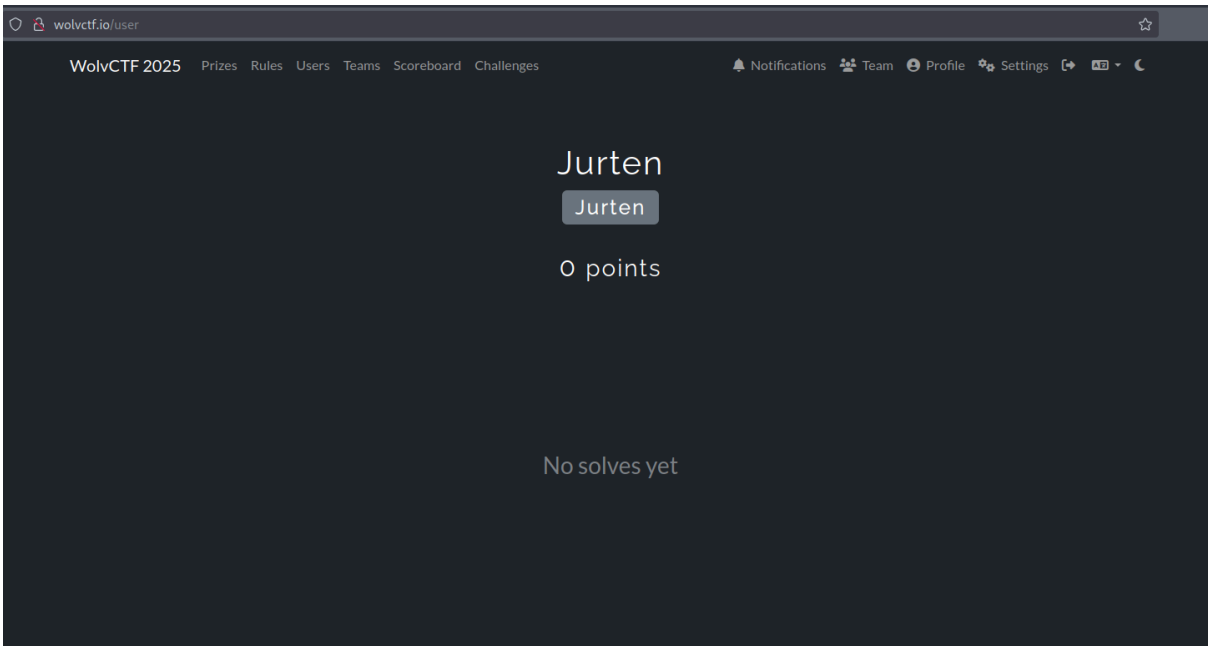
Acá vemos que el buffer almacena 32 bytes para nuestro input pero nos permite enviar un input de 64 bytes, y después hace una validación donde espera 0x42424242 que en caracteres se lee como 'BBBB', así que el payload que tenemos que armar son 64 caracteres y nuestra cadena de Bs

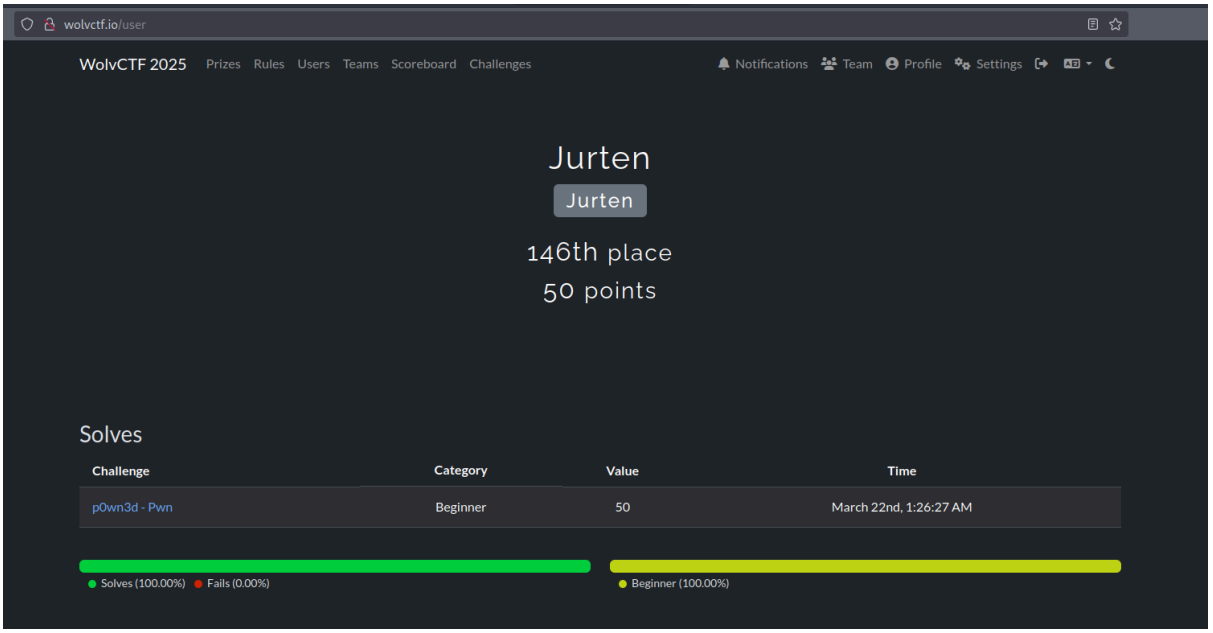
```
undefined8 main(void)

{
  char local_38 [32];
  int local_18;

  ignore();
  puts("Hello little p0wn3r. Do you have any first words?");
  fgets(local_38,0x40,stdin);
  sleep(2);
  puts("Man that is so cute");
  sleep(2);
  puts(
      "I remember last year people were screaming at the little p0wn3rs.. like AAAAAAAAAAAAA
      AAAAAAAA!"
      );
  sleep(2);
  puts("Don\'t worry little one. I won\'t let them do that to you. I\'ve set up a guard");
  if (local_18 == 0x42424242) {
    get_flag();
  }
  return 0;
}
```

La cadena que obtenemos como resultado es la siguiente: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB, y al conectarnos y enviarla obtenemos la flag

```
┌──[jurten@parrot]─[~/Desktop/wolvctf]
└─$nc p0wn3d.kctf-453514-codelab.kctf.cloud 1337
== proof-of-work: disabled ==
Hello little p0wn3r. Do you have any first words?
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
Man that is so cute
I remember last year people were screaming at the little p0wn3rs.. like AAAAAAAA
AAAAAAAAAAAAAAAAAAAA!
Don't worry little one. I won't let them do that to you. I've set up a guard
wctf{pwn_1s_l0v3_pwn_1s_l1f3}
```

flag: wctf{pwn_1s_l0v3_pwn_1s_l1f3}





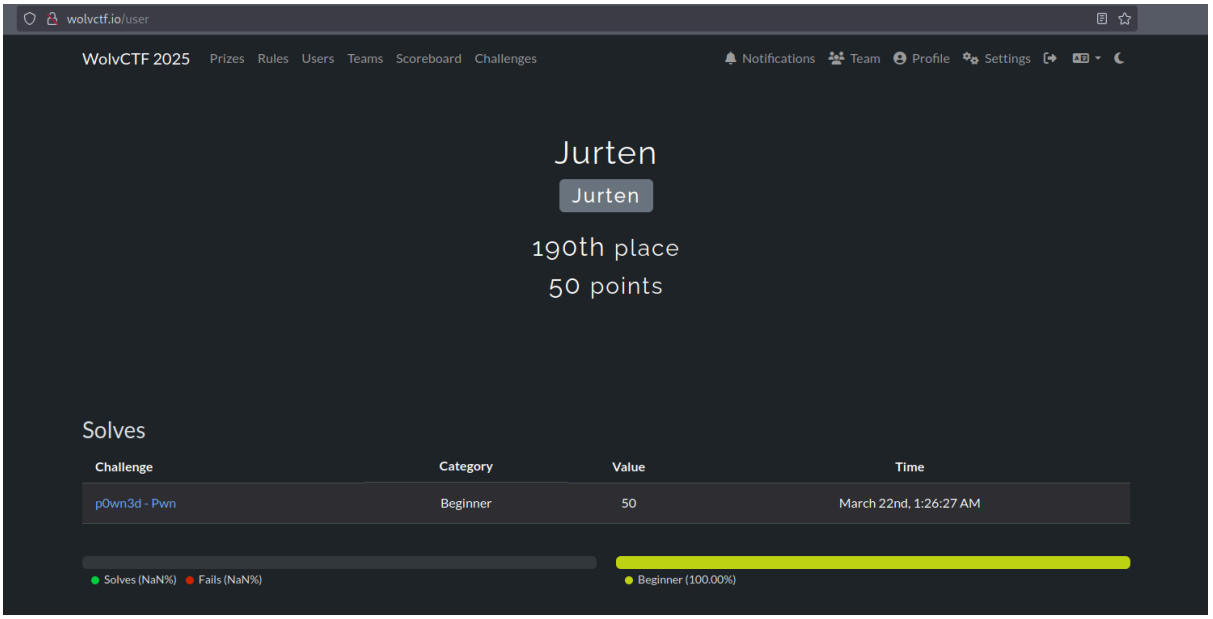
p0wn3d_2(PWN):

Este me costo un poco más, es un buffer overflow bastante clásico pero no tenia practica, al principio me costó calcular el padding y como acomodar el payload para que entre en LSB,

además lo ejecute local sin tener un archivo flag.txt en el directorio pero en fin, si vemos la imagen claramente se alocan 32 bytes para nuestro input, de ahí en adelante podemos sobreescribir las variables local_18 y local_14 (que son de 4 bytes) por los valores que van a hacer el if verdadero y así obtener la flag.

```
undefined8 main(void)

{
  char local_38 [32];
  int local_18;
  int local_14;

  ignore();
  puts("I can\'t believe you just did that. Do you have anything to say for yourself?");
  fgets(local_38,0x40,stdin);
  sleep(2);
  puts("Yeah Yeah whatever");
  sleep(2);
  puts("I\'ve got two guards now, what are you gonna do about it?");
  sleep(2);
  if ((local_18 == -0x21524111) && (local_14 == 0xbadc0de)) {
    get_flag();
  }
  return 0;
}
```

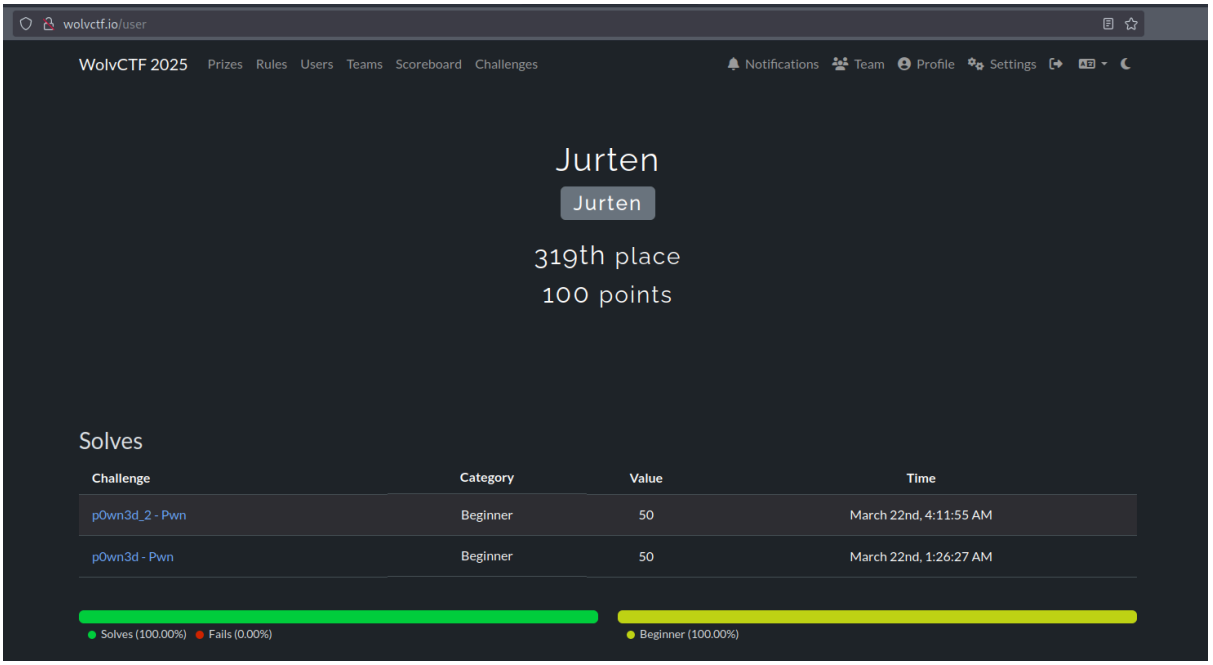
flag:wctf{4ll_y0uR_mEm_4r3_bel0ng_2_Us}



p0wn3d_3(PWN):

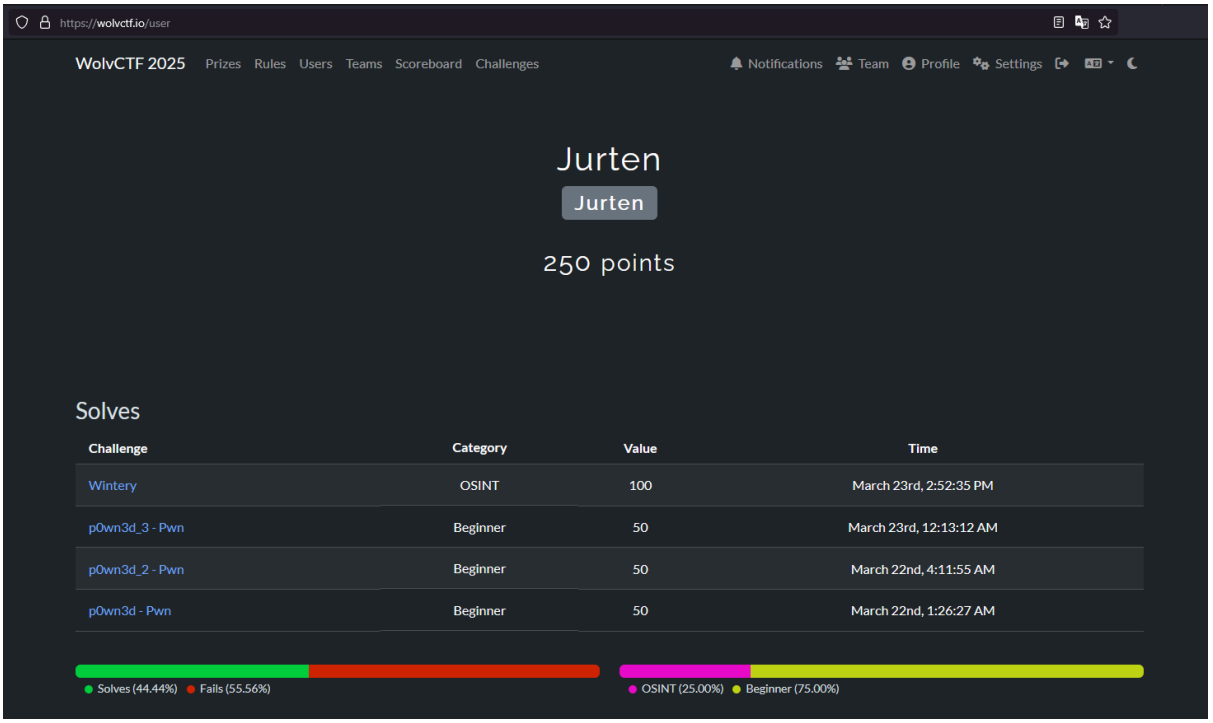
un ret2win bastante simple, utilice pwndbg para calcular el padding aunque lo tuve que hacer a mano porque cuando lo quise hacer con el comando cyclic no me tiraba respuesta, pero bueno, un pading de 40, tenemos una funcion get_flag que es la direccion que queremos poner para el retorno, que con lo hacemos con las funciones de pwntools.

flag:wctf{gr4dua73d_fr0m_l1ttl3_p0wn3r!}



Wintery(OSINT):

Estuvo muy interesante el ejercicio, desde un principio me puse a buscar herramientas que me permitieran encontrar de donde era la bandera. Primero pensé que se trataba de Umass, pero luego usando https://findpiclocation.com/ pude encontrar que se trataba de la Universidad de Michigan, con eso fue cuestión de ver mas o menos en maps qué edificios se parecía a los que veía en la foto, por la forma de los ladrillos y la disposición de las ventanas me imagine que el edificio mas cerca a la residencia era el de ‘West Quadrangle’, y luego compare las ventanas del ‘Munger Graduate Residences’ y con eso saque de google maps la ubicación y con el hint probé un par de combinaciones de los últimos dígitos hasta dar con la ubicación.
Flag: wctf{42.273,-83.741}



Me olvide de sacar la foto antes de redimir la flag pero bueno, había hecho el p0wn3d_3

Reflexiones:

Me pareció muy interesante que se pueda usar chatGPT como herramienta de OSINT porque es algo que personas no técnicas tienen fácilmente a su disposición y me pareció una llamada de atención a la información que compartimos en internet con libertad. También disfrute mucho de repasar los conceptos de la materia. Honestamente había perdido bastante contacto con lo que es ciberseguridad y sobre todo lo que es PWN me generó mucho interés y encontré muchos materiales que me ayudaron a comprender cómo funcionan estas vulnerabilidades. A pesar de todo me encontre con ejercicios de PWN que no pude resolver como lo fue DryWall de WolvCTF, que usaba seccomp para solo permitir algunas funciones y tengo la teoría de que se rompió con ROP chains y pivots pero no lo logre vulnerar.
También me generó bastante dificultad el no tener un set de herramientas para resolver los ejercicios OSINT pero creo que conseguí un set de tools que me fue suficiente y me lleve una lección de que hoy en dia no somos tan anónimos como pensamos.