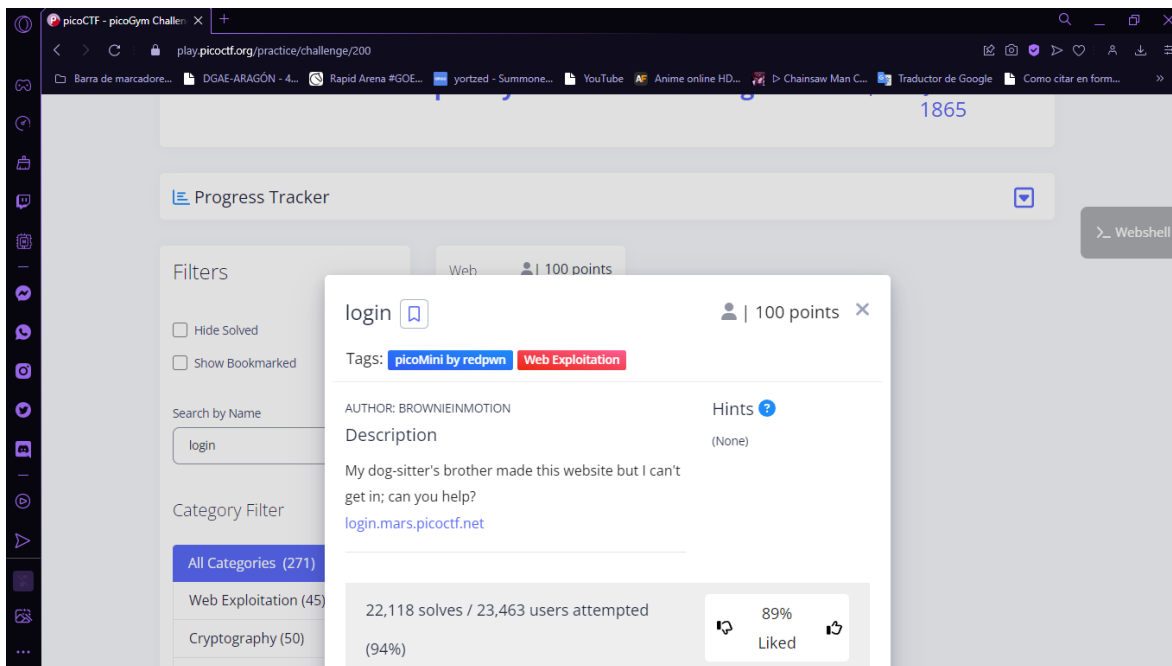


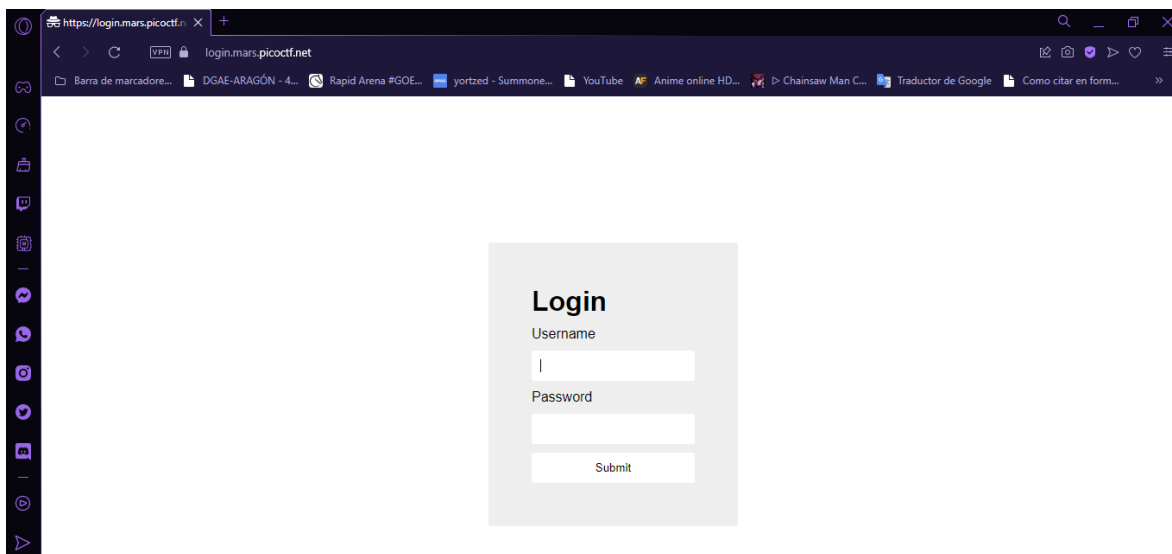
## Tarea: Login

Alumno: Roberto Carlos Quintana Escamilla

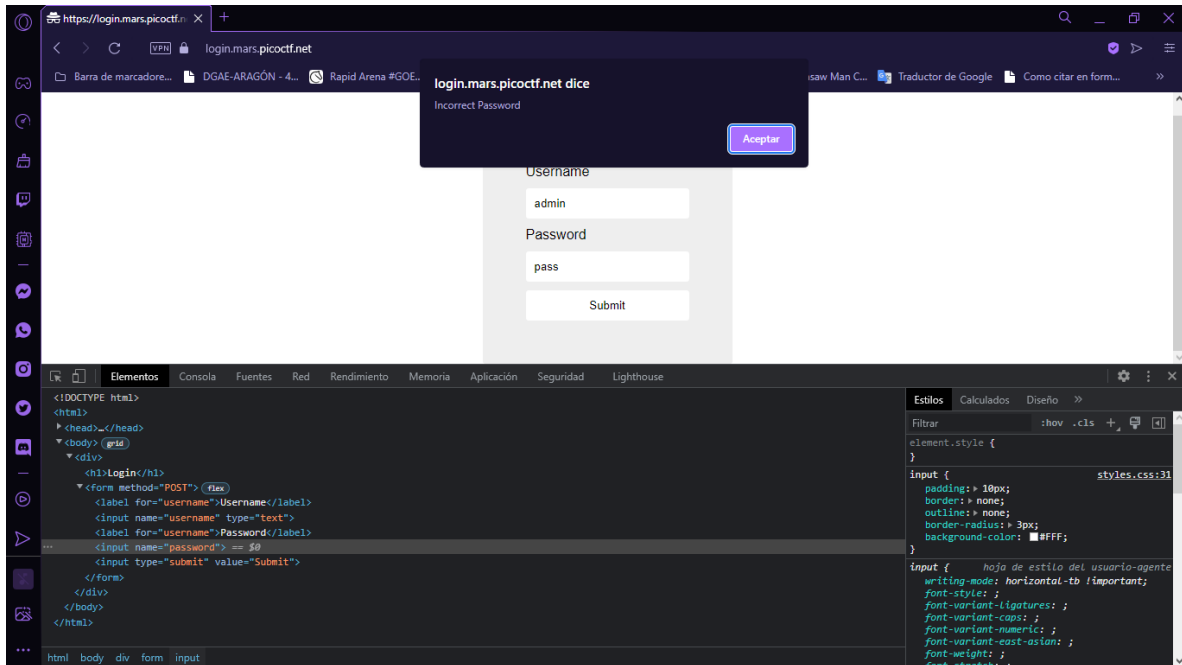
Para este ejercicio se nos pide que entremos a un sitio web creado por el cuidador de perros del hermano del solicitante (dog-sitter's brother).



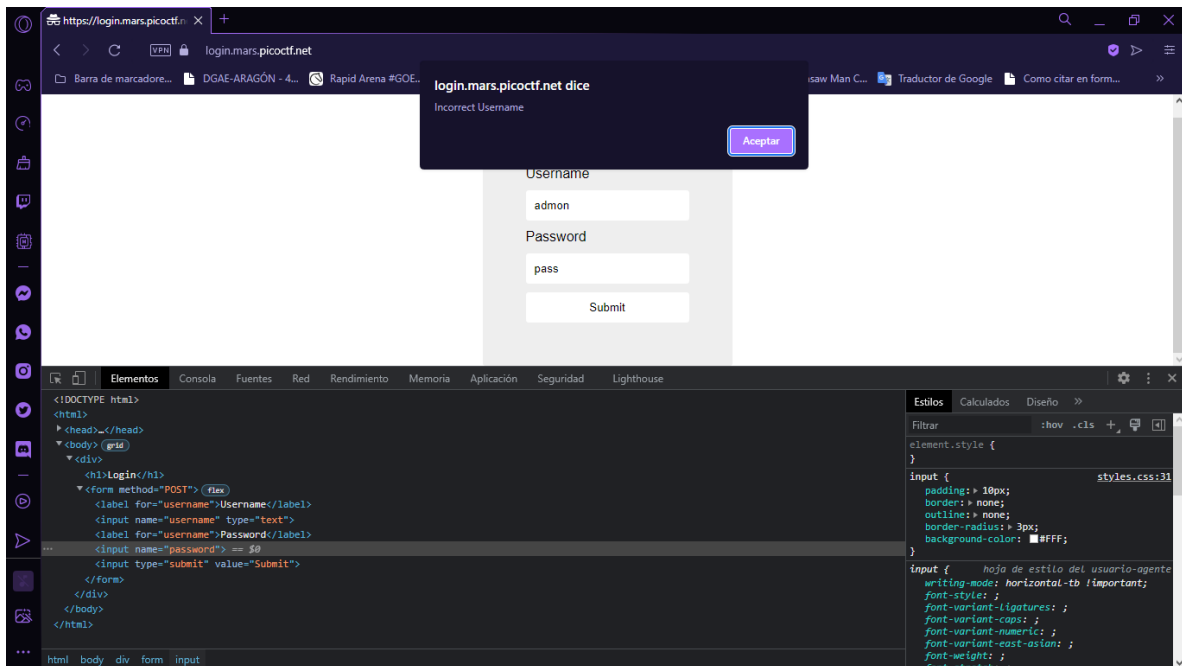
Si entramos al sitio encontramos una pagina de login simple con un texfile para usuario y uno de contraseña ademas de un boton para ingresarlos.



Si intentamos ingresar usando por ejemplo el usuario admin y el password pass (únicamente por ser comunes para este tipo de problemas) la página lanzara un alert el cual indica que la contraseña es incorrecta (se utilizo el inspector del navegador para poder visualizar el password editando la etiqueta de input).

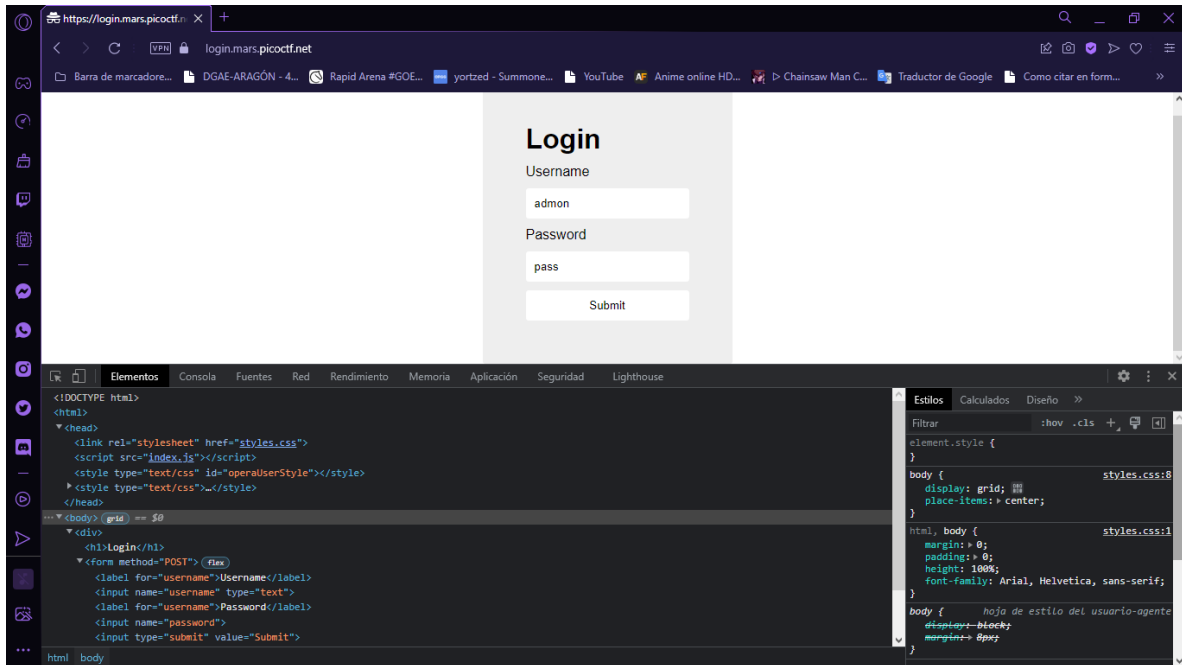


Ya que nos indica que el password es incorrecto probemos cambiar el usuario para verificar que no se lanza un mensaje genérico en caso de errar el password o la contraseña.

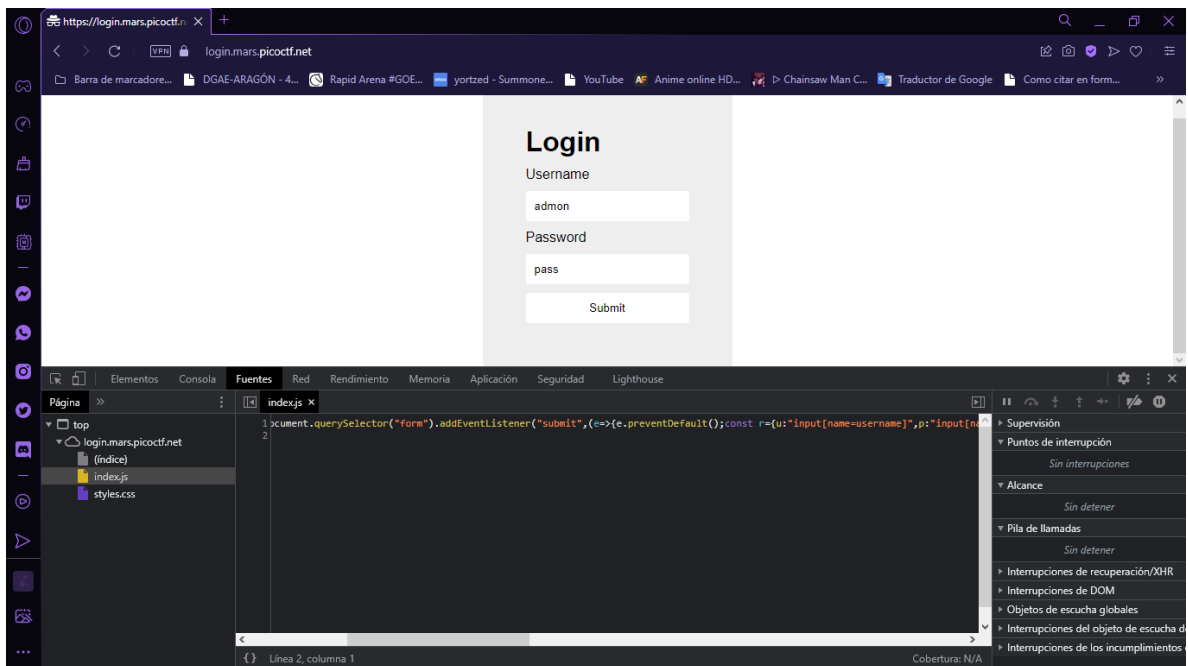


Comprobaos que no es el caso y que encontramos el usuario correcto. Inspeccionemos la función del botón submit para averiguar mas de la consulta. Para ello nos fijamos en la etiqueta del botón

correspondiente para encontrar que la función que usa es una llamada “sumit”. Luego buscamos un archivo js en el cual se puedan almacenar las funciones y encontramos index.js (esto también lo hubiéramos podido realizar simplemente verificando los archivos de la pagina con la ayuda de la pestaña fuentes).

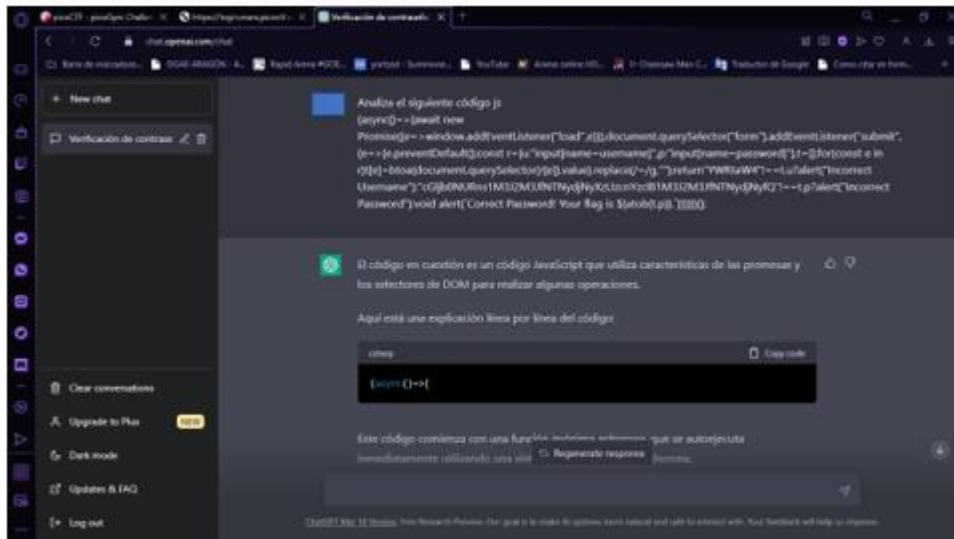


Si observamos el contenido del archivo js encontraremos una función poco entendible por la forma en la que fue escrita. Aun así podemos notar rápidamente que esta relacionada con nuestro botón sumit.



```
(async)=>{await new
Promise((e=>window.addEventListener("load",e))),document.querySelector("form").addEventListener("submit",(e=>{e.preventDefault();const
r={u:"input[name=username]",p:"input[name=password]"},t={};for(const e in
r)t[e]=btoa(document.querySelector(r[e]).value).replace(/=/g,"");return"YWRtaW4"!=t.u?alert("In
correct
Username"):"cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ"!=t.p?alert("
Incorrect Password"):void alert(`Correct Password! Your flag is ${atob(t.p)}.`)}})}());
```

Por practicidad utilizaremos la herramienta chatgpt para analizar la función con una simple consulta.



Anexo respuesta:

“El código en cuestión es un código JavaScript que utiliza características de las promesas y los selectores de DOM para realizar algunas operaciones.

Aquí está una explicación línea por línea del código:

```
(async)=>{
```

Este código comienza con una función anónima asíncrona, que se autoejecuta inmediatamente utilizando una sintaxis de función flecha asíncrona.

```
await new Promise((e=>window.addEventListener("load",e))),
```

La primera línea de esta función espera hasta que se carga completamente la página (event "load") antes de continuar con la siguiente línea de código. Esto se hace utilizando una promesa que se resuelve una vez que se ha producido el evento "load".

```
document.querySelector("form").addEventListener("submit",(e=>{
```

```

e.preventDefault();

const r={u:"input[name=username]",p:"input[name=password]"},t={};

for(const e in r)t[e]=btoa(document.querySelector(r[e]).value).replace(/=/g,"");

return"YWRtaW4"!==t.u?alert("Incorrect
Username"):"cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ"!==t.p?alert(
"Incorrect Password"):void alert(`Correct Password! Your flag is ${atob(t.p)}.`);

}))

```

a segunda línea de la función anónima establece un controlador de evento para el evento "submit" en el primer elemento de formulario encontrado en el documento. El controlador de eventos es una función que toma un evento como argumento. Dentro del controlador de eventos, se realiza lo siguiente:

Se llama al método preventDefault() del evento para evitar que el formulario se envíe y se recargue la página.

Se define un objeto "r" que contiene los selectores de los elementos de entrada de nombre de usuario y contraseña.

Se define un objeto vacío "t" que se utilizará para almacenar los valores codificados en base64 de los elementos de entrada.

Se itera a través del objeto "r" utilizando un bucle for/in y se asigna el valor codificado en base64 de cada elemento de entrada correspondiente a su clave correspondiente en el objeto "t". El método btoa() se utiliza para codificar los valores en base64 y el método replace() se utiliza para eliminar los caracteres "=" adicionales.

Se verifica si el valor codificado en base64 de "u" en el objeto "t" es "YWRtaW4". Si no es así, se muestra una alerta que dice "Incorrect Username".

Se verifica si el valor codificado en base64 de "p" en el objeto "t" es "cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ". Si no es así, se muestra una alerta que dice "Incorrect Password".

Si ambos valores son correctos, se muestra una alerta que dice "Correct Password! Your flag is <valor decodificado de p en base64>."

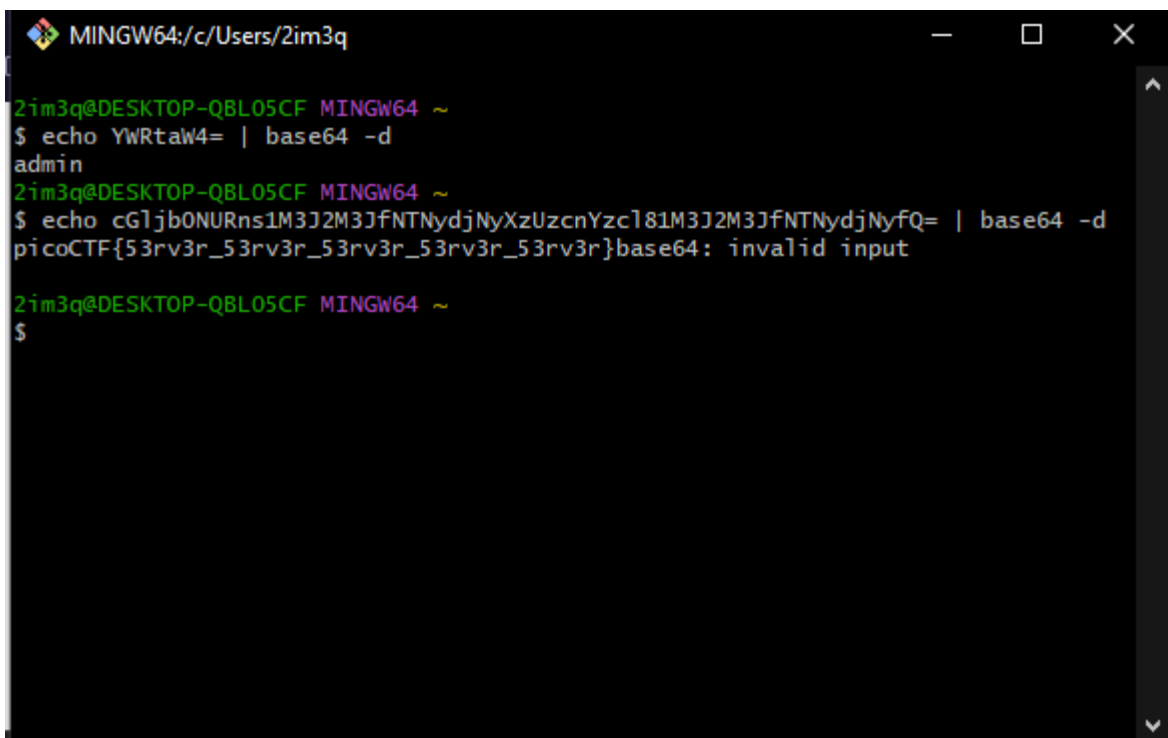
La última línea de la función anónima cierra la sintaxis de la función y la autoejecuta inmediatamente utilizando (). Por lo tanto, todo el código se ejecuta tan pronto como la página se carga completamente, y establece un controlador de eventos en el primer formulario encontrado en la página. Cuando el formulario se envía,

”

De esta consulta podemos destacar que los inputs de los texfile's se codifica en base64 y se retiran los caracteres = sobrantes, luego se comparan los resultados con "YWRtaW4=" (para el input del usuario) y con "cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ=" para el password. Por lo que solo nos hace falta agregar los caracteres = eliminados (probemos con un solo carácter al final) y decodificarlos para obtener el usuario y contraseña (aun que el usuario ya lo hemos encontrado). Para ejercitar los comandos de Linux usaremos una consola para decodificarlo con el comando base64 y el modificador -d , para proporcionar la entrada a este comando usaremos el comando echo seguido de la cadena y luego un | para utilizarlo de entrada.

UsuarioBase64 = "YWRtaW4="

PasswordBase64 = "cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ="



```
MINGW64:/c/Users/2im3q
2im3q@DESKTOP-QBL05CF MINGW64 ~
$ echo YWRtaW4= | base64 -d
admin
2im3q@DESKTOP-QBL05CF MINGW64 ~
$ echo cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ= | base64 -d
picoCTF{53rv3r_53rv3r_53rv3r_53rv3r_53rv3r}base64: invalid input
2im3q@DESKTOP-QBL05CF MINGW64 ~
$
```

Para corregir el error de invalid input de la segunda cadena añadiremos un segundo carácter =.

Nota: Si revisamos el artículo sobre base 64 en wikipedia se nos da el algoritmo concreto y sobre los símbolos igual encontramos

*"Después de codificar los datos, si en el paso anterior quedaban 2 octetos por codificar, entonces se añade el carácter '=' al final de la salida; si solo quedaba un octeto, se concatenarán dos caracteres '='. Esto avisa al decodificador de que los bits a 0 que se hayan añadido de relleno no deben formar parte de los datos reconstruidos."*

```
MINGW64:/c/Users/2im3q

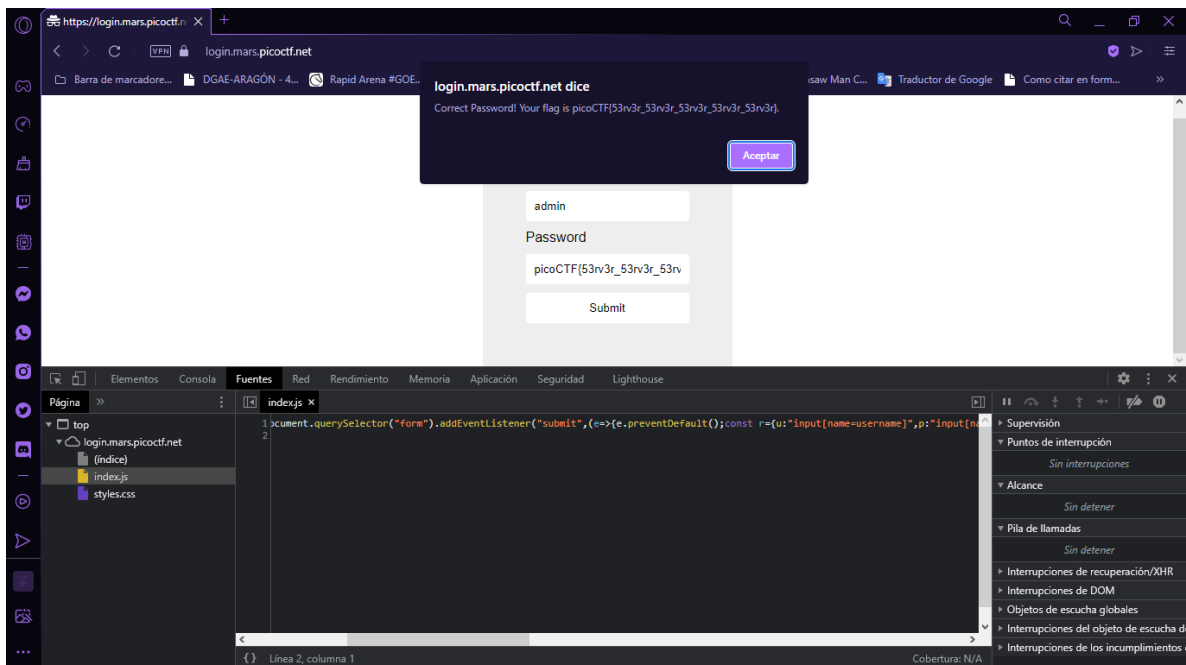
2im3q@DESKTOP-QBL05CF MINGW64 ~
$ echo cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzc181M3J2M3JfNTNydjNyfQ== | base64 -d
picoCTF{53rv3r_53rv3r_53rv3r_53rv3r_53rv3r}
2im3q@DESKTOP-QBL05CF MINGW64 ~
$
```

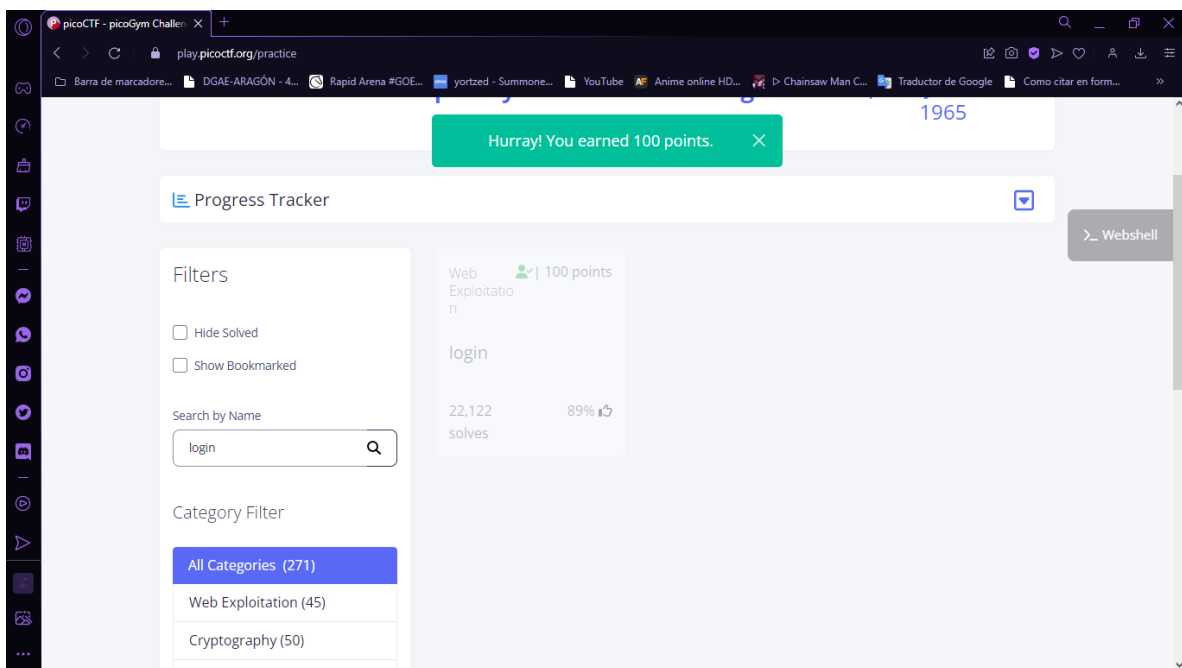
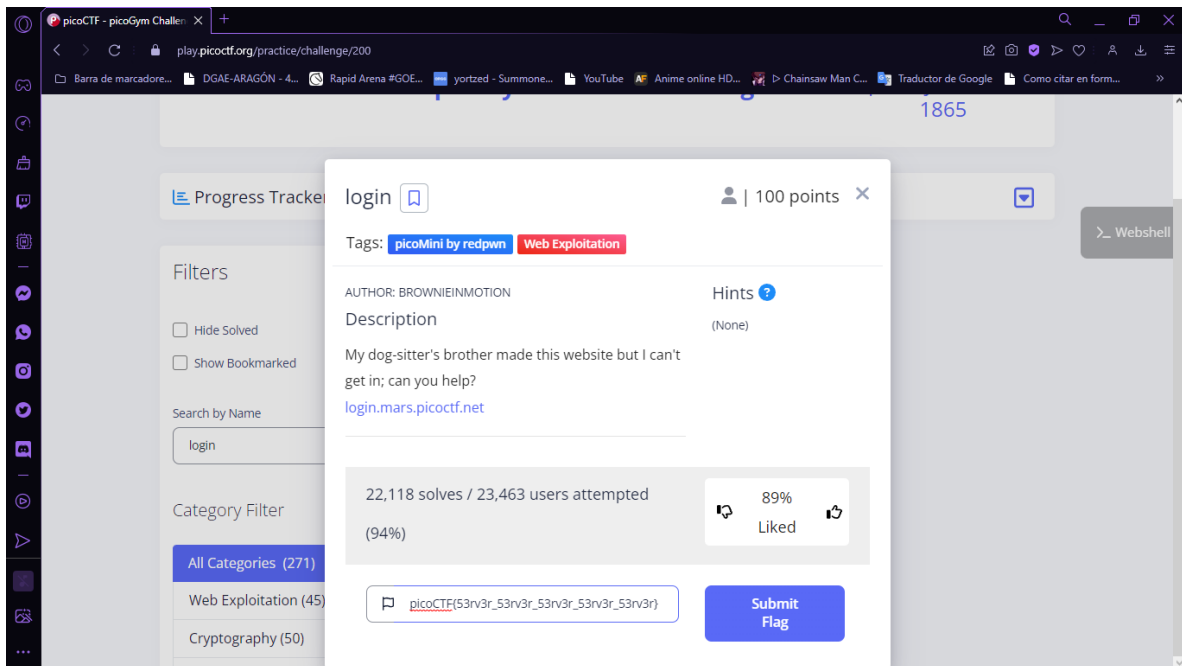
Usuario = “admin”

Pasword = “picoCTF{53rv3r\_53rv3r\_53rv3r\_53rv3r\_53rv3r}”

Flag = ” picoCTF{53rv3r\_53rv3r\_53rv3r\_53rv3r\_53rv3r}”

Comprobamos





## Vulnerabilidades

- Se está usando usuario común para realizar el Login lo cual puede facilitar el hallar las credenciales necesarias para acceder al sitio
- La verificación de credenciales se realiza en el frontend en lugar del backend lo cual da un fácil acceso a los atacantes al funcionamiento e la verificación para saber como vulnerarla.



## **Solución**

-Crear un usuario personalizado y poco común.

-Hacer la verificación desde el Backend. Para ello se requiere.

1. Configurar una base de datos para almacenar las credenciales de los usuarios. Esta base de datos deberá tener al menos una tabla para almacenar los nombres de usuario y las contraseñas encriptadas.
2. Recopilar los datos ingresados por el usuario en el formulario y enviarlos al servidor backend. Esto se puede realizar mediante el método POST.
3. En el backend, validar los datos ingresados por el usuario comparándolos con la información almacenada en la base de datos. Para hacer esto, puedes utilizar un lenguaje de programación como PHP, Python o Ruby, dependiendo del lenguaje que estés utilizando en el backend de tu página web.
4. Si los datos ingresados por el usuario son válidos, permitir el acceso a la página web restringida o a la funcionalidad específica. Si los datos son inválidos, mostrar un mensaje de error al usuario y permitirle que intente nuevamente.

## **Referencias**

- Wikipedia. (2023). Base64. En Wikipedia, La enciclopedia libre. Recuperado el 23 de marzo de 2023, de <https://es.wikipedia.org/wiki/Base64>

- Fahmida, Y (2019). Bash base64. En linuxhint. Recuperado el 23 de marzo de 2023, de [https://linuxhint.com/bash\\_base64\\_encode\\_decode/](https://linuxhint.com/bash_base64_encode_decode/)

- shwoseph (2021) .base64 -d decodes, but says invalid input. En stackexchange. Recuperado el 23 de marzo de 2023, de <https://unix.stackexchange.com/questions/631501/base64-d-decodes-but-says-invalid-input>

- ChatGPT. (2023, marzo 23). Consulta análisis de la función encontrada en el archivo index.js de la página <https://login.mars.picoctf.net> . ChatGPT. Recuperado de <https://chatgpt.com/>