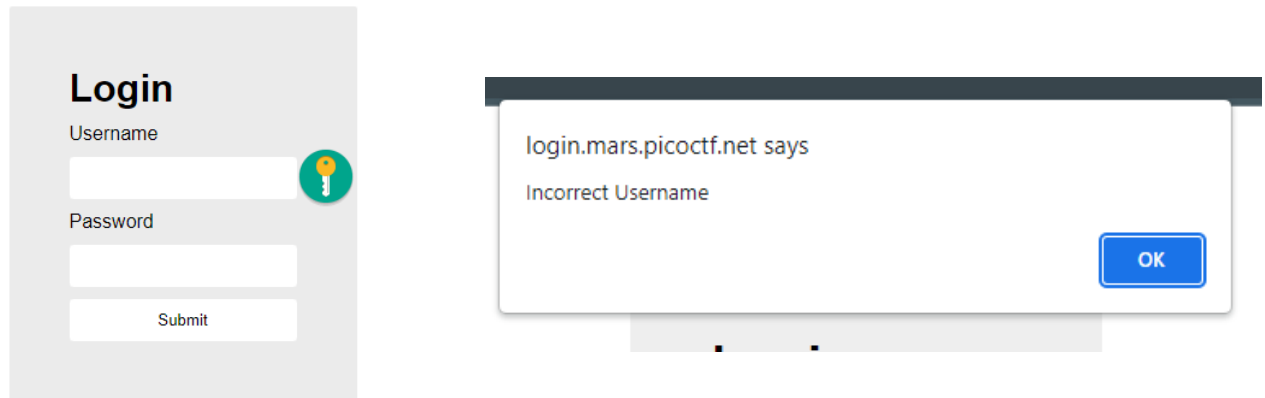


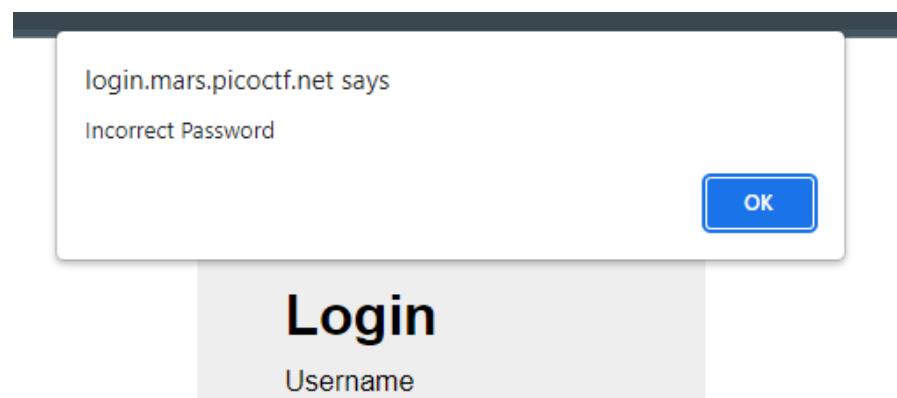
Pico CTF: login

([login.mars.picoctf.net](http://login.mars.picoctf.net))

Temos uma página simples de login que nos traz um pop up avisando se o login foi mal sucedido, o que significa que manipulação client side não é o caminho.



Tentando user admin e senha admin, o pop up muda para senha incorreta, o que significa que admin é o user correto.




Inspecionando o javascript da página, vemos as seguintes linhas de código  
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 linha grifada chama bastante atenção pois mostra o user e senha corretos, e podemos ver que eles estão criptografados, e que o alerta de login bem sucedido usa o comando atob(). atob() é usado para decodificar base64, então se decodificar o user e senha temos

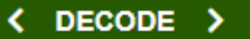

YWRtaW4

 For encoded binaries (like images, documents, etc.) use the file upload form

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries)


 Live mode OFF Decodes in real-time as you type or paste (supports

 **DECODE**  Decodes your data into the area below.

admin


user admin

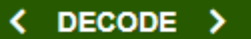

cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ

 For encoded binaries (like images, documents, etc.) use the file upload form a little

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

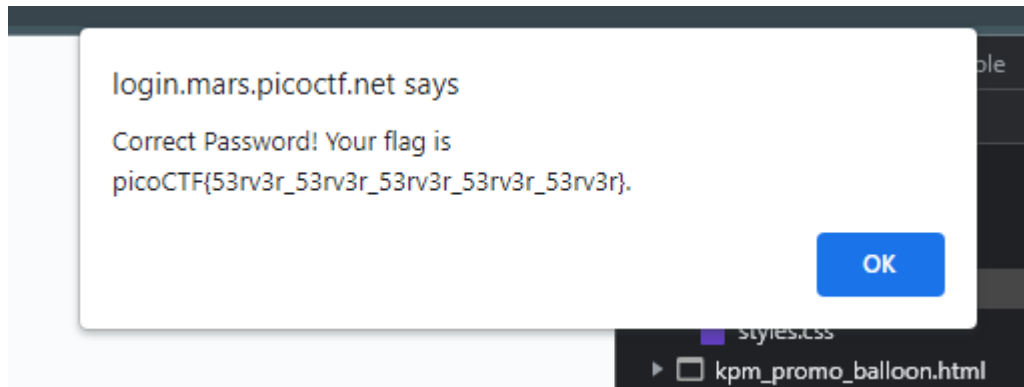
 Live mode OFF Decodes in real-time as you type or paste (supports only th

 **DECODE**  Decodes your data into the area below.

picoCTF{53rv3r\_53rv3r\_53rv3r\_53rv3r\_53rv3r}

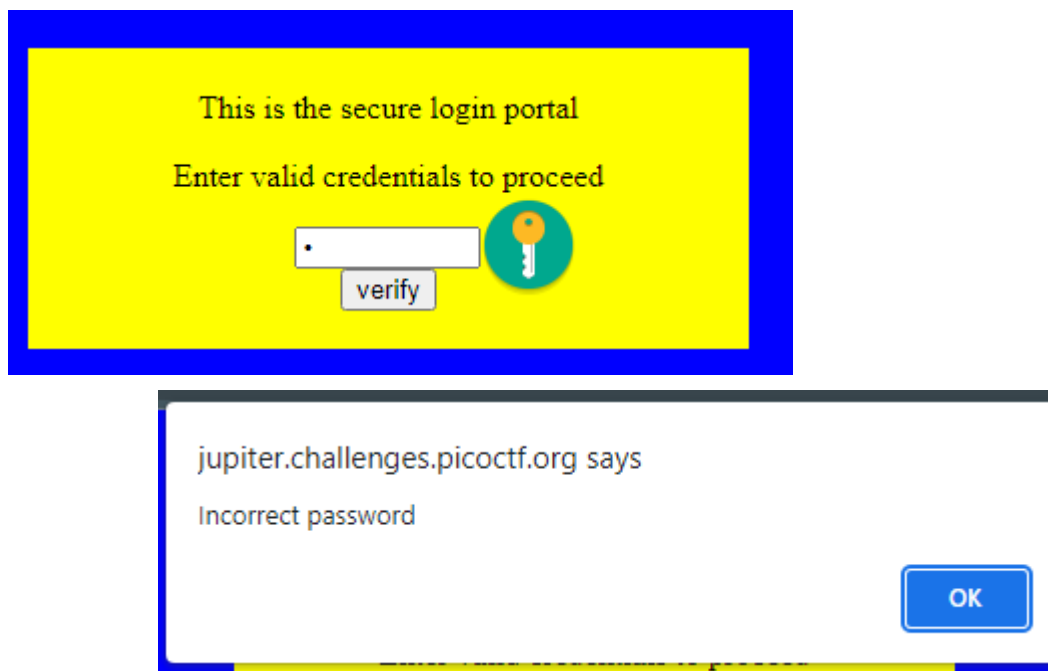
senha picoCTF{53rv3r\_53rv3r\_53rv3r\_53rv3r\_53rv3r}

aparentemente já é a resposta do CTF, mas para garantir, ao colocar o user e a senha propostos, nós temos:



E aproveitando para resolver um exercício parecido, temos o PicoCTF: dont-use-client-side (<https://jupiter.challenges.picoctf.org/problem/37821/>)

Tem uma aparência parecida ao exercício anterior



Ao inspecionar o elemento, descobrimos que é mais simples ainda o processo, pois a flag se encontra no próprio HTML:

```

checkpass = document.getElementById('pass').value;
split = 4;
if (checkpass.substring(0, split) == 'pico') {
  if (checkpass.substring(split*6, split*7) == 'a3c8') {
    if (checkpass.substring(split, split*2) == 'CTF{') {
      if (checkpass.substring(split*4, split*5) == 'ts_p') {
        if (checkpass.substring(split*3, split*4) == 'lien') {
          if (checkpass.substring(split*5, split*6) == 'lz_1') {
            if (checkpass.substring(split*2, split*3) == 'no_c') {
              if (checkpass.substring(split*7, split*8) == '9}') {
                alert("Password Verified")
              }
            }
          }
        }
      }
    }
  }
}

```

Ao encaixar os pedaços, já cada segmento mostra seu começo e fim (como em 0,split\*1 seguido de split, split\*2, etc) conseguimos a seguinte flag:

picoCTF{no\_clients\_plz\_1a3c89}

Ao colocá-la no verify, confirmamos que é a flag certa

