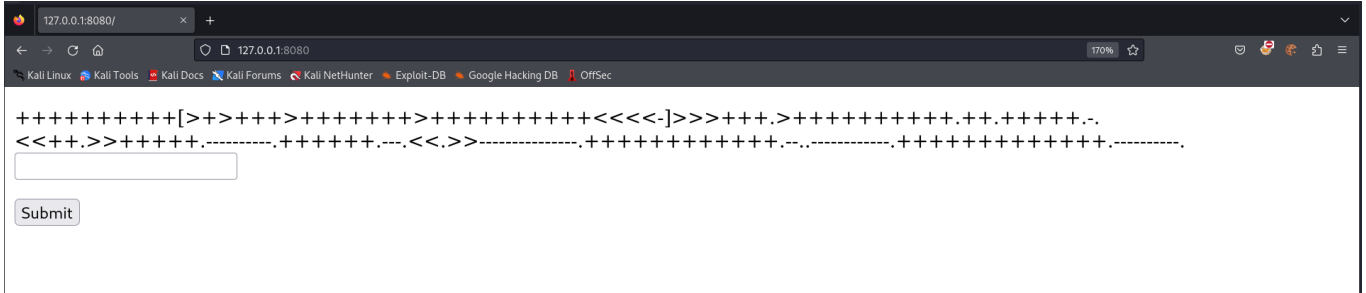
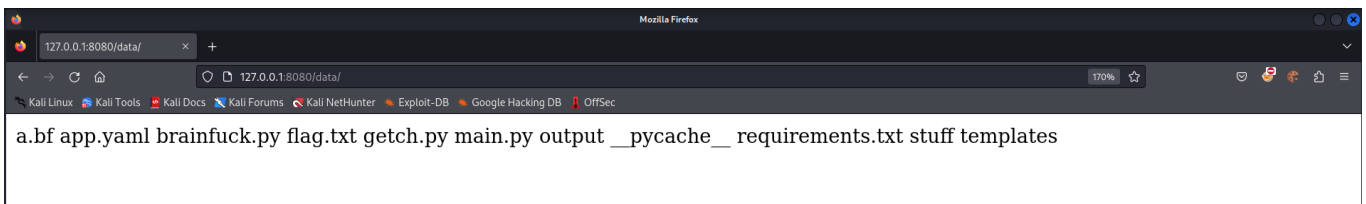
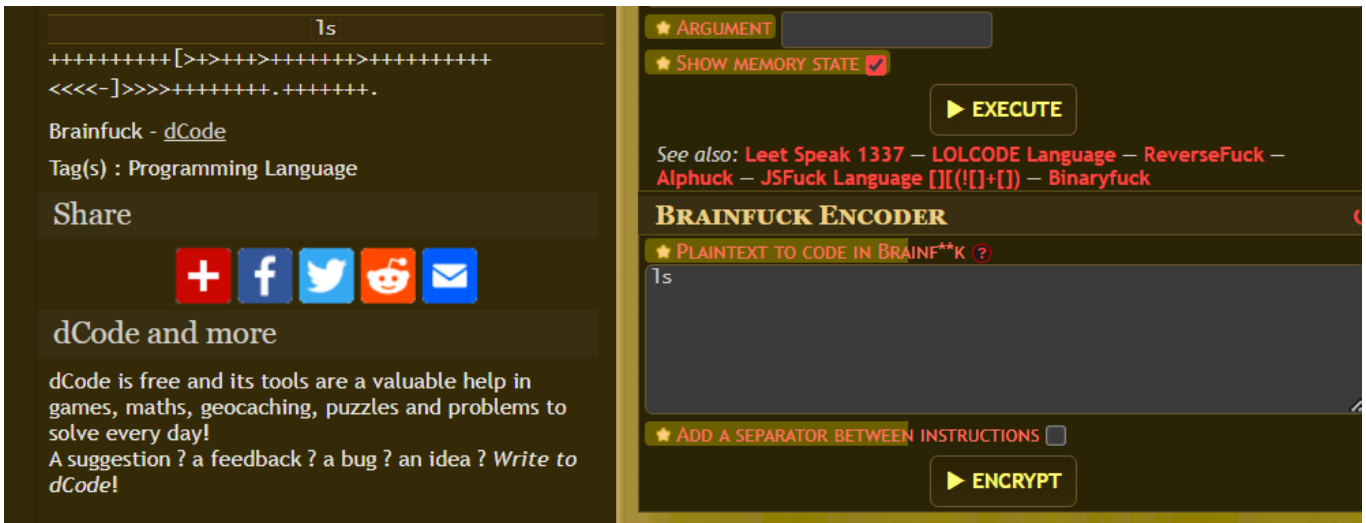


WTbF Writeup

Upon loading the webpage, the user is greeted with a piece of cipher text, followed by a submission form and a "Submit" button. The top piece of text is a piece of cipher text in "brainfuck" which, when decrypted, reads "Input your command".



After poking around, it is hoped that the user will realise they are executing commands on the web server and as such attempt to explore the current folder they are in which can be done through using a third party website to create payloads and then executing them. It is hoped that the user will uncover "flag.txt" in the directory they are currently exploring and download the file.!



Upon downloading and reading the file, there is an acronym for "GPG" which would be a hint for

the next phase of the challenge, which would be related to GPG keys.

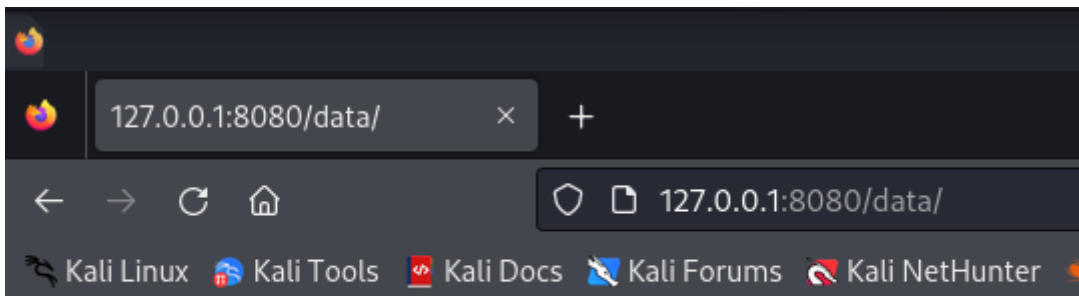
```
(kali㉿kali)-[~]
└─$ wget http://127.0.0.1:8080/flag.txt
--2023-12-15 20:31:35-- http://127.0.0.1:8080/flag.txt
Connecting to 127.0.0.1:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22 [text/plain]
Saving to: 'flag.txt'

flag.txt          100%[=====>]          22  --.-KB/s    in 0s

2023-12-15 20:31:35 (2.51 MB/s) - 'flag.txt' saved [22/22]

(kali㉿kali)-[~]
└─$ cat flag.txt
Good
Progress
Gandalf
```

As noticed in an earlier screenshot, there is a folder called "stuff". Upon trying to view the files in the "stuff" folder the user would find two files, a PGP & GPG file.



my_key.pgp real_flag.txt.gpg

The user should then download both files onto their machine. Due to the fact that we have a private key, and an encrypted message it is possible for the person to try and bruteforce the password of the private key. To start with, the user needs to convert `my_key.pgp` into a hash file,

this can be done through `gpg2john`

```
kali@kali: ~/Desktop
File Actions Edit View Help
$ gpg2john my_key.pgp > hash

File my_key.pgp
(kali@kali)-[~/temp]
$ cat hash
Johnnie:$gpg$*1*988*3072*287face9702b6aecc84f63c0ea68686cb3874702e131845d28612849f2619e70a9ad60e20fe682a7b739320705e0e4a4f4dd8fad3258c
b20c0ef10450600c431f25807d5d2f17caf8237c32d7942448b6aed6c0a777d77425131d1aa61fe622c1416bcf6ca06559a19f01f860aaf3d2fa2ade8c1b2e45056154
7d453c92e647cad00a97ae949bba3af85d3e231ba879e3a672c2430cdd52daaf17405ca8026add06ce7fcce0c5ed3927b01a3b3ceded3de65ef4821c1d4e388e735def2
7388a7cea3aeb312acaf952a3c923e4b0ba05405d9ba5cb11eafd9352aa3d098f18e91d2aa666da45e3d27ff81c5a01aa39e40677102a90913c011e29d5a917e756e
55cce97ccb7b5d8a61f241f478342c850ee2f502707b807f8392ece0662e206fb247026ae91af8f698d78698a6eaaad6a0466ac3276fcc90d2b7a8ee19c62a67de7bf7
1f49760a8dfdd95bc23295b13cd170a41384d5e8cfecd32f4809ed30f02f7d8cd638eda00498eb7b2c4a756fc77411a56c1020b1d399ac93497dd4d187201dd167b834
19caa76a19f12ff2208e943a57c7133efc1cf147f70cfc1b2df42cb0a89392635edefa36cb53ef9f4262559e99fc0b8caa135a4ef3e23ab77973949adf9354b91ab4c1
8bff2806623deb7375167aa5ec092213e499e5f46b164e29a867270601e77087d93c09ad57a59f24efc05227cef0038d3a2f34dd386c99e6f401be40f2a01b7db8c9a4
d4e556b8c358b02a0aad3e7ffa991f8dc224d26041077c9a6160829a81fcfcddeb954fbb8534b8a863599ecf8f982d1097d74945e54ee0743d2e54fb50f28c83855d45
dc724fdf6148500ff2b36246d4627a0e3e8f12d21c6b8fb9eeae0b833edea99c918ddc942c9ac45cd0aa280866cd2121b76a11576abef48362f0a2ef50a3ee83476f1
33e8d4e94a9464e3ff06016b70ba41f04581eac933eaeafd06c677b02aec265b9a87c3548682b0b30cff35a091d089b17eb191962a0cb854daf0757bb96f443cf7f5f20
e1b9dfa21937995554708c125dc4e2aea10342decf9b7fe6a4607776fc6aa01695dda131e785430b92930167f819b9284aadbd6ad6a39fc347d5904246afed714a914
3fb4c97d6efed3532fc8b69dcf1e34a4f237d4fcf29374cddb8f3ab200303821b59bc38c8ae7c396e85e8a13d6f7963b6d14c9c16ec74e7de5fc926ccd2ca7d4099e3
4f82da365d516325d07ecc07f73343a1ecf3b14e9e9193ce58e20f6bcc1ca2751f32e861c448c683c40408fcd251b4ba722494ab2beb053a344caa381cb1cc9b6133f
debb5cc45a8d170423da2ad23ff67427852686dc72f3bc019e1f349df76b0b181ae12d37790047457af0f7e728b1982300e36c2ff3d18d0b89d2cc6c0b137*3*254*2*
7*16*1e8a621c4635f5eb06e824a66e9bca1e*65011712*634c66a915452021::: Johnnie <try@harder.hacksoc>::my_key.pgp

(kali@kali)-[~/temp]
$
```

This "hash" file can now be passed into JohnTheRipper with the `rockyou.txt` wordlist and upon running this command, would output the password for the private key.

```
(kali@kali)-[~/temp]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (gpg, OpenPGP / GnuPG Secret Key [32/64])
Cost 1 (s2k-count) is 65011712 for all loaded hashes
Cost 2 (hash algorithm [1:MD5 2:SHA1 3:RIPEMD160 8:SHA256 9:SHA384 10:SHA512 11:SHA224]) is 2 for all loaded hashes
Cost 3 (cipher algorithm [1:IDEA 2:3DES 3:CAST5 4:Blowfish 7:AES128 8:AES192 9:AES256 10:Twofish 11:Camellia128 12:Camellia192 13:Camellia256]) is 7 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
123456789 (Johnnie)
1g 0:00:00:00 DONE (2023-12-15 20:39) 7.692g/s 30.76p/s 30.76c/s 30.76C/s 123456..password
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

The user can then import the GPG key and decrypt the other file they found, revealing the real flag for this challenge.

```
(kali㉿kali)-[~/temp]
```

```
$ gpg --import my_key.pgp
```

```
gpg: key 813D3C79ACF9C2BC: "Johnnie <try@harder.hacksoc>" not changed
```

```
gpg: key 813D3C79ACF9C2BC: secret key imported
```

```
gpg: Total number processed: 1
```

```
gpg:          unchanged: 1
```

```
gpg:      secret keys read: 1
```

```
gpg:  secret keys unchanged: 1
```

```
$ gpg --decrypt real_flag.txt.gpg > output
```

```
gpg: encrypted with 3072-bit RSA key, ID 2C9DAD3E3F9BFFD6, created 2023-12-15
```

```
my_R... "Johnnie <try@harder.hacksoc>"
```

```
(kali㉿kali)-[~/temp]
```

```
$ cat output
```