Python Source Virus and Antivirus Tutorial

Please read all of this assignment closely before you begin!

This assignment is designed to reinforce basic concepts about how viruses work. Please follow the naming/calling conventions used in this document exactly as you write your code. This assignment will count as two homework grades. You will submit your source code electronically as two email attachments. Please include your name as part of each filename, e.g., **`smithVirus.py`** and **`jonesAntivirus.py`**. Please send the final version of your code to cs235homework@gmail.com. Questions should be sent to my Wofford email address, sloanjd@wofford.edu. You may work on this individually or in pairs as you see fit. But there should be no discussion of this assignment with anyone in the class before submission with the exception of your partner when doing pair programming.

You will be writing both a source code virus for Python files and antivirus software designed to detect and remove your virus. This tutorial will step you through the process. The virus you write will not be particularly virulent or dangerous provided you stay within the bounds of this assignment. Nonetheless, you should do all your coding on a personal computer and not on any of Wofford's computers. For real virus research, you would use an isolated machine used only for this purpose. For this project, just keep all your code in a separate directory. And, yes, your code must be written in Python for rather obvious reasons.

Writing the Virus:

Please put this code in a single file.

1. You will be writing a virus that "infects" Python source files. That is, it will copy itself into a Python source file. This will be an appending virus so *all* the code will be added at the end of the file. This will make it particularly easy to add or remove. The first line of your virus will be a signature

   ```
   ## "#@! infected by virus !@#"
   ```

   Yes, it is a bit simplistic that the signature is so easily spotted, but this whole assignment is a bit simplistic.

   Write a function **`infected()`** that takes a file name as its only parameter. It should return **`True`** or **`False`** depending on whether the file contains the signature. Notice that the file that contains **`infected()`** would give a false positive if checked with your code. Think about how you might fix but do not attempt to do so.

2. Write a function **`selectTarget()`** that has no parameters. It should open the current working directory, retrieve a list of all files in that directory, filter out any non-Python files (files that don't have a "**`.py`**" extension), and any already infected files. (You'll use **`infected()`** to determine this. Randomly select one of the remaining files. Return that file's name. Return **`False`** if there are no such files.

   There are several commands in the **`os`** module that you will probably find useful.

   ```
   import os                ## loads os
   os.getcwd()              ## returns the current directory
   os.listdir(d)            ## returns a list of files in d
   ```

3. Write a function **`copyCode()`** that takes a file name as an parameter. This function will open two files, the file passed as a parameter will be the target to be infected and will be open for appending. You will write the viral code to the end of this file.

   You will also need to open the current file (**`__file__`**) for reading so you can search through it for the virus (conveniently located at the end of the file). Read and ignore each line in this source file up to the line with the viral signature. Everything from this point on will be appended to the target file.

4. Modify **`copyCode()`** so that immediately after writing the virus signature it inserts an additional comment that specifies the name of the file that is the source of the infection and the name of the file being infected. The first few lines of an infection might look like:

   ```
   ...
   ## "#@! infected by virus !@#"
   #/Users/guest/Desktop/Virus/virus.py infected four.py

   import os
   ...
   ```

   This is just a precaution.

5. Write a function **`payload()`**. This should simply print a one-line message of your choice. This is the viral payload.

6. Write a function **`infect()`** with no parameters. It should call **`selectTarget()`** to identify a target. If a target exists, it should call **`copyCode()`** with the target. If no target exists, it should occasionally call **`payload()`**, say with a probability of 1 in 4.

(Deciding when to run **`payload()`** is the viral trigger. We could create more elaborate tests if we wished.)

7. Add the line

   ```
   infect()
   ```

   to the bottom of your code so the code will run automatically when loaded.

Your virus is complete. Test it only on *your own* files and computer.

AntiVirus Code:

Now we will write the antivirus code. Open a new file for this code.

1. First you will need code to determine if a file is infected. You have already written this. Copy **`infected()`** into this file.

2. Write a function **`scan()`**, with no parameters, that scans the current directory for infected files. It should print out the name of any file that is infected. You'll likely want to use code from the **`selectTarget()`** function you have already written. Don't worry about false positives.

3. Write a function **`removeVirus()`** that take a file name as a parameter. It should remove the viral code from this file. Open the file for reading and record each line up to the line with the signature. This is the original source code. Everything else is the virus. Close this file and reopen it for writing. Write out the recorded code. *Were that it was really this easy to remove a real virus!*

Ideally, you could modify **`scan()`** so that it automatically calls **`removeVirus()`** but we will not do this since many of our functions contain the viral signature and our **`infected()`** function will give false positives.

Discussion:

This is a simplistic example but you have now seen how the basic components of viral and antiviral code are written. There are lots and lots of additions we could make—more elaborate tests to trigger the delivery of the payload, more elaborate payloads, encryption, scanning the entire filesystem for targets, etc. And, we should be working with machine code. Also, we used a very explicit signature that made the coding quite easy.

False positives are a concern of any antivirus code, particularly if you want to do any repairs to files. Our antivirus code will flag itself as infected. This problem comes up when running multiple antivirus programs on the same computer.

Because we restricted our code to looking in a single directory, looking only at Python files, limited the payload to printing a message, and made no effort to hide the code, this is *relatively* harmless code. But I strongly recommend that you keep this in a directory that is separate from any other Python code you are writing. And if you plan on keeping the code, you might save it in a compressed format.