Python PE Parser

Objectives

Learn how to write a Python Program to Analyze a Windows PE File.

Part #1: Installing and Configuring Python Virtual Environment
Part #2: Write a Python Program to Analyze a Windows PE File

Overview

By now you have an idea what a Windows Portable Executable (PE) file is and used a couple of tools to analyze a PE file. This time we will see how we can benefit from our Python skills to write our own PE analyzer.

Requirements & Resources

The steps below have been tested to work with Python 3.

- Use your ThreatActor system (Kali Linux)
- pefile: https://github.com/erocarrera/pefile
- Introduction: https://github.com/erocarrera/pefile/blob/wiki/UsageExamples.md
- Examples: https://www.programcreek.com/python/example/91048/pefile.PE
- More pefile examples can be found here: http://www.programcreek.com/python/example/50993/pefile.DIRECTORY_ENTRY
- Download the packchecker.py file.

Part #1: Installing and Configuring Python Virtual Environment

Let's start by installing virtualenv and the pefile module.

1. Download and install virtualenv, which can be done using the following:
   $ sudo apt update
   $ sudo apt install python3-pip
   $ pip3 install virtualenv
   $ vim .bashrc
   → add the virtualenv to the PATH
   PATH=$PATH:/home/kali/.local/bin
   export PATH
   $ source .bashrc

2. Create a virtual environment for your work using virtualenv as below:
   $ virtualenv pework

```
kali@kali:~$ virtualenv pework
created virtual environment CPython3.8.4.final.0-64 in 362ms
  creator CPython3Posix(dest=/home/kali/pework, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/kali/.local/share/virtualenv)
    added seed packages: pip==20.1.1, pkg_resources==0.0.0, setuptools==44.0.0, wheel==0.34.2
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
kali@kali:~$
```

3. Now after you finished creating a virtualenv, let us activate it and install pefile.
   $ source pework/bin/activate

4. You should now be within the **pework** environment and should see something like this:

```
kali@kali:~$ source pework/bin/activate
(pework) kali@kali:~$
```

5. Then to install the pefile module, all we need to do is:
   $ pip install pefile

```
(pework) kali@kali:~$ pip install pefile
Processing ./.cache/pip/wheels/42/52/d5/9550bbfb9eeceaf0f19db1cf651cc8ba41d3bcf8b4d20e4279/pefile-2019.4.18-py3-none-any.whl
Processing ./.cache/pip/wheels/8e/70/28/3d6ccd6e315f65f245da085482a2e1c7d14b90b30f239e2cf4/future-0.18.2-py3-none-any.whl
Installing collected packages: future, pefile
Successfully installed future-0.18.2 pefile-2019.4.18
(pework) kali@kali:~$
```

6. To test that everything is successful, just open a python interactive interpreter and run:
   >>> import pefile

```
(pework) kali@kali:~$ python
Python 3.8.4 (default, Jul 13 2020, 21:16:07)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pefile
>>>
```

7. If you get no errors, then exit() python and continue.

Part #2: Write a Python Program to Analyze a Windows PE File

In this assignment we want to write some python code in order to analyze our PE files with the help of the pefile module we installed.

1. First start your python interpreter again.
2. The first thing to do is import the pefile module:
   ```
   import pefile
   ```

3. Now let's analyze the file klogger.exe, so we need to load it using pefile. This can be done like this:
   ```
   pe = pefile.PE('/usr/share/windows-binaries/klogger.exe')
   ```

4. Now suppose we want to know the address of the Image Base. We can use the following:
   ```
   hex(pe.OPTIONAL_HEADER.ImageBase)
   ```

5. What if we wanted to know the address of the Entry Point? This could be done like this:
   ```
   hex(pe.OPTIONAL_HEADER.AddressOfEntryPoint)
   ```

6. Also, if you want to know the number of sections in this PE file, we can do:
   ```
   hex(pe.FILE_HEADER.NumberOfSections)
   ```

7. Okay, that seems nice right ☺? Now let us print all the sections found in the file. This could be done like this:
   ```
   for section in pe.sections:
   ... print (section.Name, \
         hex(section.VirtualAddress), \
         hex(section.Misc_VirtualSize), \
         section.SizeOfRawData )
   ```

Now time to do some work alone, without my help ☺
Deliverable #1: Show how to dump all the information of a PE file?

Deliverable #2: Show how to check if the file is an EXE or DLL?

Deliverable #3: Write some code to parse the IAT.

Deliverable #4: How can you change the Address of the Entry Point to 0xBEEFBEEF?

Suppose you want to check if the file is Packed or not, you can use the packchecker.py file given. It can be used like this:
packchecker.py putty.exe

Deliverable #5: What is the packchecker code checking for and why (hint: read the code and try to understand it)? Explain your answer.

Deliverable #6: Please reflect on your learning from this lab and what was not clear to you so we can discuss it together.

DIY: Now write the changes to a new file called "putty.exe" and try solving some of the questions we used in our PE lab, but instead of using CFF Explorer, use your python skills ☺