

5.1 Automation - BlueJupyter

We have already come a long way. In Cyber Security we all should know how to automate things just to save time so that we can focus on other important tasks.

Now, we have another challenge, we have lots of malware samples (Windows Based) and we need to run the initial analysis on them.

Scenario is as below.

Analyst,

I'm -REALLY- sorry about this, but I have samples to triage and I won't be able to get to them today. There are only a few of them (26 total LMAO). Can you handle them for me? Thanks!





-Other Analyst

<https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/5-1.Automation-BlueJupyter>

So, in such cases we can have things automated using our favourite programming language. But HuskyHacks has provided us with the samples and jupyter's notebook to save much of our time.

Downloading The Samples

The samples are provided in the GitHub Repo.

main		PMAT-labs / labs / 5-1.Automation-BlueJupyter /
husky release v1.0		
..		
	.gitkeep	release v1.0
	FORTRIAGE.7z	release v1.0
	README.md	release v1.0
	password.txt	release v1.0

We need to have them downloaded and placed in the *malware-samples/dropbox* directory.

```
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis$ ls
dropbox  Malware-Analysis.ipynb  MalwareSample.py  __pycache__  README.md  saved-specimens
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis$ cd dropbox/
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis/dropbox$ ls
out0.exe  out12.exe  out15.exe  out18.exe  out20.exe  out23.exe  out2.exe  out5.exe  out8.exe
out10.exe  out13.exe  out16.exe  out19.exe  out21.exe  out24.exe  out3.exe  out6.exe  out9.exe
out11.exe  out14.exe  out17.exe  out1.exe   out22.exe  out25.exe  out4.exe  out7.exe
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis/dropbox$
```

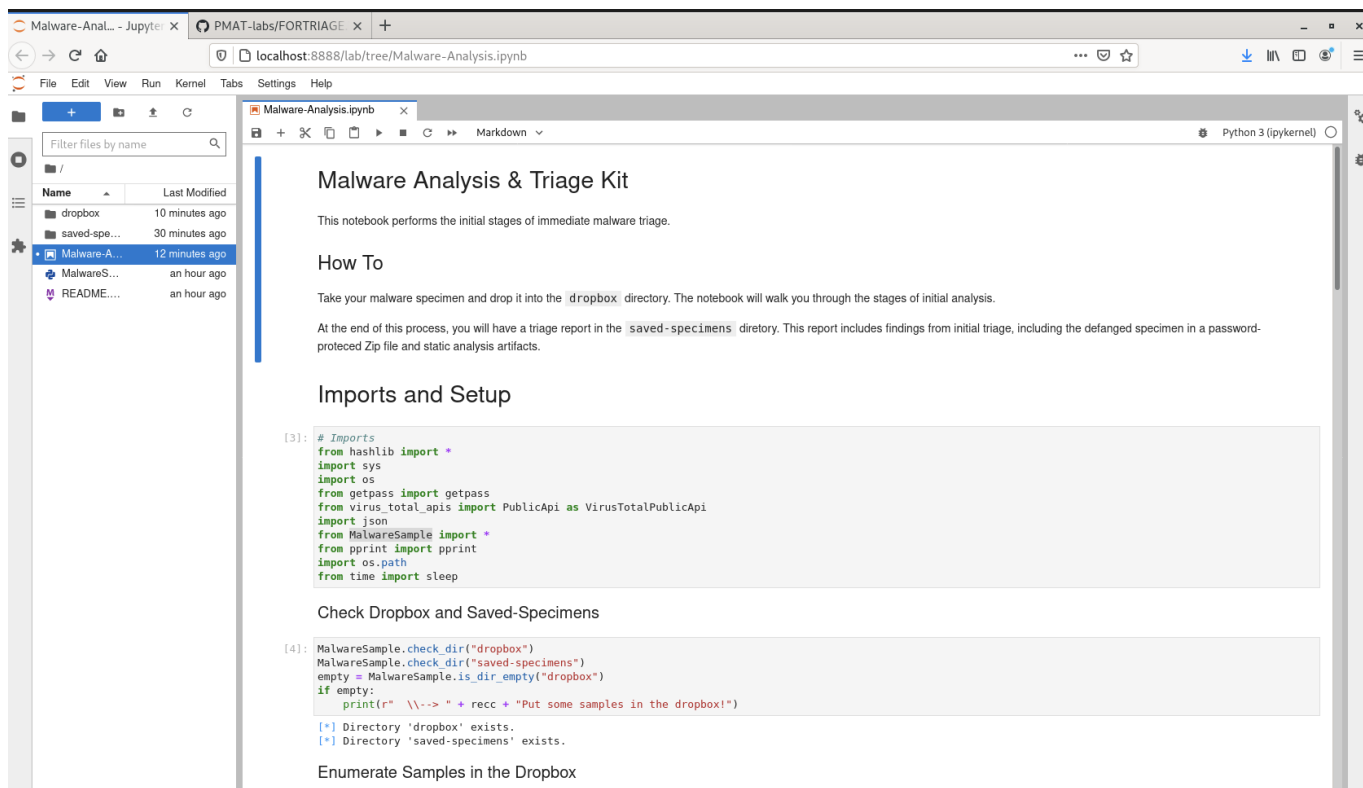
Once we have them in place, we need to move ahead.

Running Jupyter-Lab

Run the following command on your terminal to start the Jupyter-Lab.

```
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis$ jupyter-lab Malware-Analysis.ipynb
[I 2022-03-22 14:01:42.923 ServerApp] jupyterlab | extension was successfully linked.
[I 2022-03-22 14:01:42.930 ServerApp] nbclassic | extension was successfully linked.
```

On success, you'll be greeted with the following page.



Stepping Through The Notebook

There are a lot of things going on in this notebook. So, we will begin with the initials. At first, like in every programming language,

```
# Imports
from hashlib import *
import sys
import os
from getpass import getpass
from virus_total_api import PublicApi as VirusTotalPublicApi
import json
from MalwareSample import *
from pprint import pprint
import os.path
from time import sleep
```

The second code snippet is about checking the directories whether they exist or not.

Check Dropbox and Saved-Specimens

```
[4]: MalwareSample.check_dir("dropbox")
      MalwareSample.check_dir("saved-specimens")
      empty = MalwareSample.is_dir_empty("dropbox")
      if empty:
          print(r"  \\--> " + recc + "Put some samples in the dropbox!")

[*] Directory 'dropbox' exists.
[*] Directory 'saved-specimens' exists.
```

The third code snippet is about checking and printing the names of the malware samples.

Enumerate Samples in the Dropbox

```
[12]: samples=!ls dropbox/*  
      for s in samples:  
          print(info + "Sample: " + s)  
  
[*] Sample: dropbox/out0.exe  
[*] Sample: dropbox/out10.exe  
[*] Sample: dropbox/out11.exe  
[*] Sample: dropbox/out12.exe  
[*] Sample: dropbox/out13.exe  
[*] Sample: dropbox/out14.exe  
[*] Sample: dropbox/out15.exe  
[*] Sample: dropbox/out16.exe  
[*] Sample: dropbox/out17.exe  
[*] Sample: dropbox/out18.exe  
[*] Sample: dropbox/out19.exe  
[*] Sample: dropbox/out1.exe  
[*] Sample: dropbox/out20.exe  
[*] Sample: dropbox/out21.exe  
[*] Sample: dropbox/out22.exe  
[*] Sample: dropbox/out23.exe  
[*] Sample: dropbox/out24.exe  
[*] Sample: dropbox/out25.exe  
[*] Sample: dropbox/out2.exe  
[*] Sample: dropbox/out3.exe  
[*] Sample: dropbox/out4.exe  
[*] Sample: dropbox/out5.exe  
[*] Sample: dropbox/out6.exe  
[*] Sample: dropbox/out7.exe  
[*] Sample: dropbox/out8.exe  
[*] Sample: dropbox/out9.exe
```

The fourth code snippet is about creating the directories for each malware sample.

```
MalwareSample('dropbox/out3.exe')
```

```
[6]: sample_obj = [MalwareSample(s) for s in samples]
```

Create a Saved Specimen directory for the specimen(s)

```
[7]: for obj in sample_obj:
      saved_sample_name = MalwareSample.create_specimen_dirs(obj.sample_name)
      obj.saved_sample_name = saved_sample_name
```

The sixth code snippet is about defanging the sample files so that they become non-executable or un-executable.

Defang Sample

```
[8]: for obj in sample_obj:
      sample_path = MalwareSample.move_and_defang(obj.sample_name, obj.saved_sample_name)
      obj.sample_path = sample_path
```

```
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis/saved-specimens$ cd 03-22-2022-140520_
03-22-2022-140520_out0.exe/          03-22-2022-140520_out22.exe/
03-22-2022-140520_out10.exe/        03-22-2022-140520_out23.exe/
03-22-2022-140520_out11.exe/        03-22-2022-140520_out24.exe/
03-22-2022-140520_out12.exe/        03-22-2022-140520_out25.exe/
03-22-2022-140520_out13.exe/        03-22-2022-140520_out2.exe/
03-22-2022-140520_out14.exe/        03-22-2022-140520_out3.exe/
03-22-2022-140520_out15.exe/        03-22-2022-140520_out4.exe/
03-22-2022-140520_out16.exe/        03-22-2022-140520_out5.exe/
03-22-2022-140520_out17.exe/        03-22-2022-140520_out6.exe/
03-22-2022-140520_out18.exe/        03-22-2022-140520_out7.exe/
03-22-2022-140520_out19.exe/        03-22-2022-140520_out8.exe/
03-22-2022-140520_out1.exe/         03-22-2022-140520_out9.exe/
03-22-2022-140520_out20.exe/        03-22-2022-140520_SampleNegative.txt/
03-22-2022-140520_out21.exe/        03-22-2022-140520_SamplePositive.txt/
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis/saved-specimens$ cd 03-22-2022-140520_out10.ex
e/
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis/saved-specimens/03-22-2022-140520_out10.exe$ l
s
Malware.out10.exe.malz  sha256sum.txt
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis/saved-specimens/03-22-2022-140520_out10.exe$
```

The seventh code snippet is about calculating the hashes and saving them in files.

SHA256 Sum

```
[16]: for obj in sample_obj:
      hash = MalwareSample.get_sha256sum(obj.sample_path, obj.saved_sample_name)
      obj.sha256sum = hash
      print(info + obj.sample_name + ": " + obj.sha256sum)

[*] out0.exe: b56983e0168c49413ae8af56ed9a96f3e0cc2f584a3f1c985aacf8f609057582
[*] out10.exe: 7d9415b6a84c8e60e15491eb3a1d9a2831682ae07f993ab46c0aa70377098320
[*] out11.exe: 2bca8171c2d97fcc5275108af744833c5d8899f7527267f45b4f38bcf3fb3348
[*] out12.exe: f172c6ba8e0198f39d3acc07034e772c432fe63877435de7837ee56dc0bd7f9e
[*] out13.exe: c71095163dd0f96cdd11d3cd351988961e783567c29c7a2a48ad03f08c82b19b
[*] out14.exe: b0172ef49187e0988f2f0c50027e70dd0b6828f27b8f1a0f77ede415b854d8b3
[*] out15.exe: ea40bb474274c8ea7d030afd6b6d26f15993b20878857492d5eafe1c477c3650
[*] out16.exe: c2bc187d302d29e25dacd8034ca1dce97ea3b547c6a3d0087adfca31c618d519
[*] out17.exe: 305fee352917721c645b1c19c98d10df4c63b52053389453bbfa50fee1e4aad7
[*] out18.exe: 0622de553b2afdc9f204554bddaf51213c0bdfe38dafd390f69c6d83e114b7c8
[*] out19.exe: 131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbfd8267
[*] out1.exe: 9e6508ae048a5a60dd751e61616412ee648479e38c16dca6a36f5730870bef38
[*] out20.exe: 7bfd619de6029ccf480c245ae861a74aeae7abae42fdc160ab8e8818ab4ab8e6
[*] out21.exe: c6d3166aa8bf663e3d3898cf3f59cfca5e7cf130fa881b010453327ee7f56679
[*] out22.exe: 300029d33e768e64437d7758db0148c161b41fe6c630bfa9bc531141dd8f6a10
[*] out23.exe: 624e7a311c20a3e7b76d82a019339d1585d8ef8d3f052951c298a0a6be81d522
[*] out24.exe: 56739d7582bf3da59e077ae410937ec337ce867bea630ce14d3cd57bdcfce40c
[*] out25.exe: 27ba0cdbecf5919b4ad3541e5890f67d1b78014df81676723cf24fc6c04f3766
[*] out2.exe: 87e8a30af1bbc10330f972422b179e1b5520be7fbb5aa31c8f7570d8286f2b64
[*] out3.exe: 43f9756db5d91e18624e54f3c94210c6981c707cee3a29393016690a7a4cf055
[*] out4.exe: f93f3c19f0ba90b085ee72e639b056ad55aa3260a9ef6419bfce30c8ce86821d
[*] out5.exe: a436ae8beb1742b40c432ab1c2b42f62bdea17f079ccc7e147cbda1d4139ab4a
[*] out6.exe: e172b44167bd8ef0ad209a123db1500827ace22a2d55d8bed531475d190b3a53
[*] out7.exe: f98df4489487914673b36426f9a6ce41c2c0d04cd7578e4d2bbd40b55fa490a7
[*] out8.exe: 9f85833fcad0087b1aa88feb3051c7a6f88870a0c4e9f73bd1ff29d1868d4393
[*] out9.exe: 90783908ac985a6406ef00c3e882c2c8528e2b96b722512353f3630880ea98ec
```

```
s
Malware.out10.exe.malz sha256sum.txt
remnux@remnux:~/PMAT/blue-jupyter/malware-analysis/saved-specimen
at sha256sum.txt
7d9415b6a84c8e60e15491eb3a1d9a2831682ae07f993ab46c0aa70377098320r
/malware-analysis/saved-specimens/03-22-2022-140520_out10.exe$
```

The eighth code snippet is about utilizing StringSifter developed by FLARE Team to hunt down the strings based on the characters length provided and the strings will be written to a file for later analysis.

StringSifter

StringSifter is a FLARE developed tool that uses an ML model to rank a binary's strings by relevance to malware analysis.

```
[19]: length = int(input(recc + "Input your desired minimum string length [default is 4, 6-8 is recommended] > "))

[*] Input your desired minimum string length [default is 4, 6-8 is recommended] > 6

[20]: for obj in sample_obj:
    MalwareSample.pull_strings(length, obj.saved_sample_name, obj.sample_path)
```

```
[*] Written to outfile: saved-specimens/03-22-2022-140520_out0.exe/StringSifter-Out.log
[*] Written to outfile: saved-specimens/
[*] Written to outfile: saved-specimens/
[*] Written to outfile: saved-specimens/
[*] Written to outfile: saved-specimens/
[*] Written to outfile: saved-specimens/
6.22,The specified thread is not detached
6.21,2.2.14
6.21,2.2.14
6.21,tRPL6<
6.20,Net connection reset
6.20,Connection timed out
6.20,Bad address
6.20,DSO load failed
6.20,Stale NFS file handle
6.20,mswsock
6.19,Disc quota exceeded
6.19,min avg max
6.19,The given path is relative
6.19,Directory not empty
6.19,File name too long
6.19,Message too long
6.19,Bad file number
6.19,Request too long
6.18,RSHWPhe
6.16,KMGTFE
```

The ninth code snippet is about utilizing VirusTotal API to send and retrieve reports and Maliciousness of the files.

VT Analysis

Submit samples to Virus Total and generate a malicious confidence level.

```
[ ]: VT_API_KEY = getpass("Enter VirusTotal API Key (blank if none): ")

[ ]: if VT_API_KEY:
    vt = VirusTotalPublicApi(VT_API_KEY)
else:
    print(info + "No VT API Key. Skipping...")
```

Note: If there are more than 4 samples in the dropbox, hashes are submitted with a sleep of 16 seconds to remain under the public API rate limit. So hit go, grab a beverage of choice, stretch out and relax. This could be a while depending on how many samples you're submitting.

```
[ ]: if VT_API_KEY:
    for obj in sample_obj:
        print(info + obj.sample_name + ":")
        print(r" \--> " + info + "SHA256sum: " + obj.sha256sum)
        res = vt.get_file_report(obj.sha256sum)
        conf = malicious_confidence(res)
        print(r" \--> " + info + "Confidence level: " + str(conf))
        crit_level = determine_criticality(conf)
        obj.criticality = crit_level

        if len(sample_obj) >= 5:
            sleep(16)
else:
    print(info + "No VT API Key. Skipping...")
```

Finally we are making the samples password protected and removing the unprotected samples.

Zip and Password Protect

```
[*]: for obj in sample_obj:
      zip_file = MalwareSample.zip_and_password_protect(obj.sample_path, obj.saved_sample_name)
      MalwareSample.delete_unzipped_sample(obj.sample_path, zip_file)
```

Debug Object Vars

```
[*]: for obj in sample_obj:
      pprint(vars(obj))
```

One thing that we need to notice here is that most of the functions are in `MalwareSample.py` because the author has made a module, in which different functions and their functionalities are defined. Thus, we don't see the bigger picture and only have the bird's eye view.

So, you got to take a look into that file to understand more onto how to write your own python modules to complete such tasks.

Follow Me

Twitter: <https://twitter.com/deFr0ggy>

GitHub: <https://github.com/deFr0ggy>

LinkedIn: <https://linkedin.com/in/kamransaifullah>