Configuration Manual Report

By,

Ajay Kumar Oad

22183841

Table of Contents:

Chapter 1#: Process of Creating Dataset 1

List of tools used:

Table 1 List the tools and technologies used to create dataset 1.

Sr No#	Tool Name	Tool Type	Version	Reason
1	Virus Total API	Code integration	V3	To get the required information in a more automation way due to large dataset
2	Python Script	Lunix Terminal	3.11.2	To automate the downloading of Json files using API integration
3	Visual Studio	SDK	2022	To write C# code for data extraction from Json files and transformation
4	VS code	SDK	1.85	To interpret Json files
5	VirtualBox	VM	7.0.12	To run python code on Linux environment

Step 1: API Identification & Parameters Setup

Our initial step involved pinpointing the relevant VirusTotal API. This API required specific parameters file identification (SHA-256, SHA-1, or MD5), relationship type ('behavior'), a limit on related objects (set at 40), and an API key for authentication.

(See here: https://virustotal.readme.io/reference/files-relationships)

The API required specific parameters for a successful run. These included:

- **id:** A string type value representing the file's identification (SHA-256, SHA-1, or MD5).
- **relationship:** A string denoting the relationship type, such as 'behavior' in this case.
- **limit:** An int32 type representing the maximum number of related objects to retrieve, set at 40 for this extraction.
- **x-apikey:** A string type representing the user's API key necessary for API access and authentication.

Step 2: Python Code Integration (API Access)

To interact with the identified API, a Python script was developed, encapsulating these parameters. This script successfully accessed the VirusTotal API, retrieving behavioral details for the 215 fileless malware samples.

REQUEST

```python
import requests

url = "https://www.virustotal.com/api/v3/files/97245fb75db5406f2da92b4163c816a863d613e9d491be8adbee8a34f0779ac2/behaviours?limit=40"

headers = {
    "accept": "application/json",
    "x-apikey": "b361b3b0adf131ee28d2f9c1efa5e6b507a84e592e02e736c7ef58d5460b1b29"
}

response = requests.get(url, headers=headers)

print(response.text)
```

Figure 1 shows a sample of API call in Python Language

Note: Figure 1 shows, the sample of so it will run for one malware, but for the automation for collection (more than one) of hashes see my code below:

```
import os
import json
import time
import requests
import hashlib

apikey = 'ec4c77311a53e45452a305f5884a8525e9a4550d4f33ca9cedbf3356d8d53593'
hashes_file_path = "/home/ajayoad/Desktop/getbehav/hasheslist.txt"
output_directory = "/home/ajayoad/Desktop/getbehav/jsons/"

VTlink = "https://www.virustotal.com/gui/file/"

with open(hashes_file_path, "r") as hashes:
    for hashn in hashes:
        hash_value = hashn.strip()
        print('Checking hash ' + hash_value)
        url = f"https://www.virustotal.com/api/v3/files/{hash_value}/behaviours?limit=40"
        headers = {
            "accept": "application/json",
            "x-apikey": apikey
        }

        response = requests.get(url, headers=headers, timeout=120)

        if response.status_code == 404:
            print(f"Hash {hash_value} not found in VirusTotal Database")
            analysis_result = {
                "Link": VTlink + hash_value,
                "Result": "Not Found in VirusTotal Database"
            }
        elif response.status_code == 200:
            result = response.json()
            analysis_result = {
                "Link": VTlink + hash_value,
                "Result": result
            }

        # Save the response to a JSON file with the name of the current hash value
        json_file_name = hash_value + ".json"
        json_file_path = os.path.join(output_directory, json_file_name)
        with open(json_file_path, "w") as json_file:
            json.dump(analysis_result, json_file, indent=4)

        time.sleep(1 * 20)  # Sleep for 20 seconds between API requests
```

Additionally, you can also find complete code with required files for this code, here. [source-code/ Python code for getting Behaviours - Dataset 1].

Step 3: C# Code Development (Behavioural Data Processing)

After getting the Json files of 215 fileless malwares, the next process is to extract the required features (14 API-based) from those Json files.

- ➔ To do this we can use the code at [source-code/DatasetFilteration]. This project is written in Visual studio, we need to have a proper environment to run this project. Please refer to Table 1.
- ➔ Make sure you set paths in your system to run the project.
  For example, following paths needs to be updated at following lines of code.

```
Line-26: folderPath = @"G:\Thesis\Dataset-1\VT-jsons-API\jsonfiles\"; //change the path to yours:

Line-59:hashFilePath= @"G:\Thesis\Dataset-1\VT-jsons-API\samplehashes.txt";//change the path to yours:

Line-174: using (StreamWriter writer = new StreamWriter(@"G:\Thesis\Dataset-1\VT-jsons-API\VTdataset1.csv"))
```

The project also has a class diagram file, which shows the detailed version of the utilization of different classes of above project.

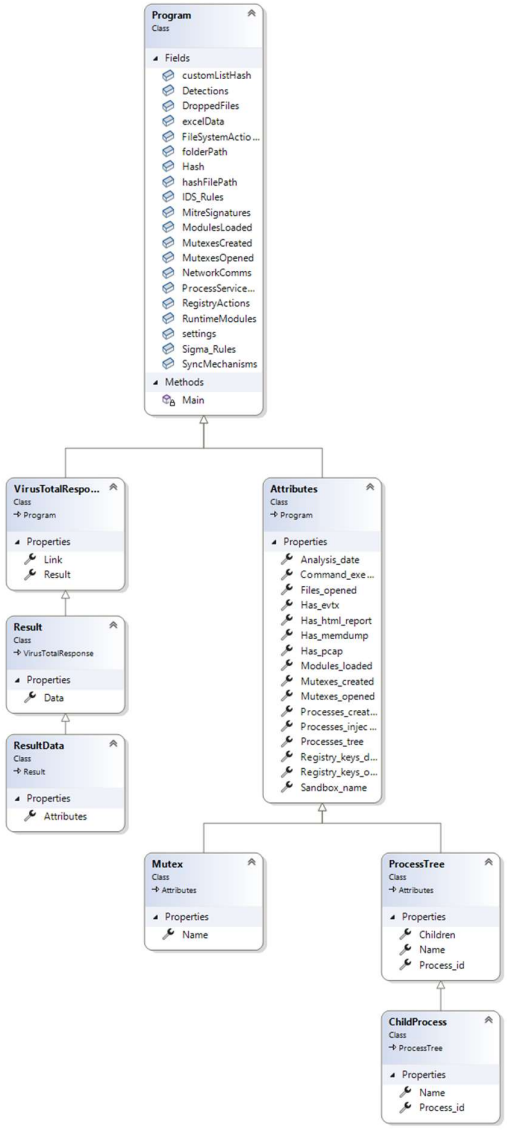

Figure 2 shows the class diagram of C# project for creating dataset 1.

Step 4: CSV Compilation

Once the above 3 steps have been followed carefully, you will see the required dataset 1 created at path which you mentioned in Line 174 of above code.

Chapter 2#: How to do Plotting (AV Detection)?

List of Tools used:

Sr No#	Tool Name	Tool Type	Version	Reason
1	Virus Total API	Code integration	V3	To get the required information in a more automation way due to large dataset
2	Python Script	Lunix Terminal	3.11.2	To automate the downloading of Json files using API integration
3	Visual Studio	SDK	2022	To write C# code for data extraction from Json files and transformation
4	VS code	SDK	1.85	To interpret Json files
5	VirtualBox	VM	7.0.12	To run python code on Linux environment
6	CSV	Data analysis	Office 365 (version: 2311)	To do the Statistical analysis such as Mean, Std, Min and Max.

Step 1. Selecting API

To get the detection rate we had to malware samples for example in dataset 1 we created above, Dataset 2 we already had access to, and dataset 3 was downloaded. For more details, please refer to 3.1 Section of my report.

Now that we have the malware we want their detection rates, to do so we need to do it again automation with API integration. But this time we needed another type of API called as "Get a file report." Please see the documentation for that here: https://virustotal.readme.io/reference/file-info

Step 2. Integrating with Python

Like creating dataset 1, we used python script, in this process we did the same with little different code. You can find the complete code in this path [Source-code/Python code for getting Plotting - Dataset 1]

Please follow following to run the python script and to see the required results:

➔ Make sure you have changed the following paths to your system locations.

```
hashes_file_path = "/home/ajayoad/Desktop/ploting/hasheslist.txt"
output_directory = "/home/ajayoad/Desktop/ploting/jsons/"
```

➔ To Add the hashes of dataset which you want to create AV detection, for example if you want to create for Dataset 1, then you must add the Dataset 1 hashes into **haeshlist.txt** files see the path [Source-code/Python code for getting Plotting - Dataset 1/**haeshlist.txt**]

Once above done then you can run the python script as following command:

python ploting.py

➔ This will generate Json files with required ifnomation in folder called jsons , in this path [Source-code/Python code for getting Plotting - Dataset 1/**jsons**]

Step 3: Extracting the features:

Now that we have Json files, we need to extract the required information iteratively and accurately. To do this we have a C# based project given at [Source-code/**Ploting**].

Note to run this project successfully you need to change the paths to your own system locations.

Following locations need to update before running project.

```
Line 12:  public static string folderPath = @"G:\Thesis\Dataset-1\Ploting\VT-Jsons-full-report\";

Line 32:  public static string hashFilePath = @"G:\Thesis\Dataset-1\Ploting\samplehashes.txt";

Line 111: using (StreamWriter writer = new StreamWriter(@"G:\Thesis\Dataset-1\Ploting\ploting.csv"))
```

These can be found in Program.cs files inside project.

Step 4: CSV Compilation:

Once we have done step 3 correctly, we will extract the file in your location if the code line 111 has been changed.

Step 5: Data Analysis:

Once we have created the CSV now, we can do the AV detection analysis on the data using CSV data analysis feature.

To do so, follow these steps:

➔ Goto data -> data analysis
➔ Select Description Statistics

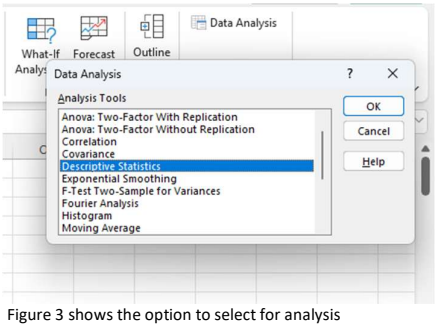

Figure 3 shows the option to select for analysis

➔ Next, we will see another window, we you need to select the input columns which are the 3 generated columns in csv, but the numeric area must be selected as input.

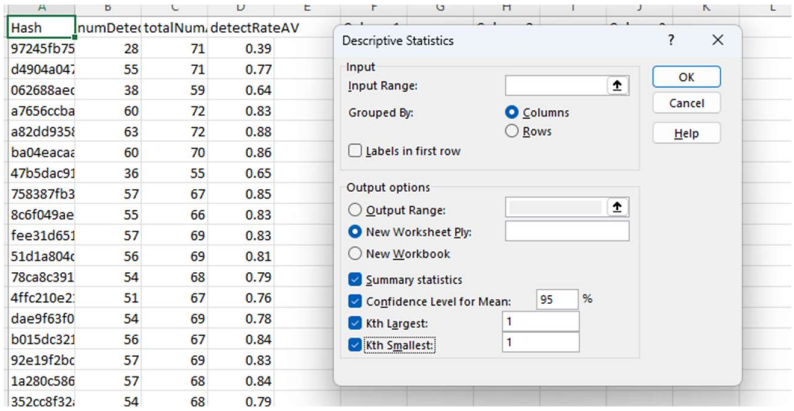

Figure 4 shows the window with input and out fields to be selected.

Also make sure we have selected the options select in blue options above Figure 4.

➔ Once we set input and output, we click ok and will see the AVD detection results, as seen in Figure 5 below.

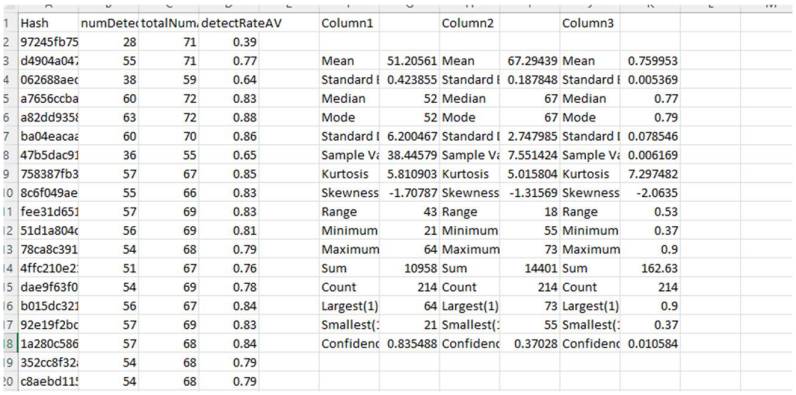

Figure 5 shows the AV detection results.

Chapter 3#:How to do ML classification?

List of tools used:

We used Weka tool with 3.8.6 version.

Step 1: Selecting file.

We can select the required dataset CSV file from the path Dataset-2 or Dataset-3.

Step 2: Selecting options.

Open Weka and click Explorer and select open file, now select any CSV from above path as mentioned in Step 1.

Step 3: Doing Classifications

Once file upload removes the field with Category/Name or other which has string and select Label/Class *(as both datasets has different label names)* and select required classifier and run the Weka test.