Report B6 Malware Intra - Family Variability

This is report is done to actually explain to how this tools was build in order to analyze a group of malware in many different aspects. The aspects that was taken into was file size, entropy, different similarity hashes including ssdeep and tlsh, virus total match, compilation time, imphash and matching yara signature.

Throughout the report I will explain on how a tool was build for each of the aspects and a sample output will be also given for a better understanding on how each tools works.

- File Size of Malware Samples (filesize.py)

For this aspect I have a created a tool that can actually each analyze each sample of malware and generate a pie chart that can be used to see how many malware in percentage falls in a file size range category.

For the file size range category I have set which is a small size below 10000 bytes or a medium size which falls somewhere between 10000 and 30000 bytes and quite large file which is beyond 30000 bytes.

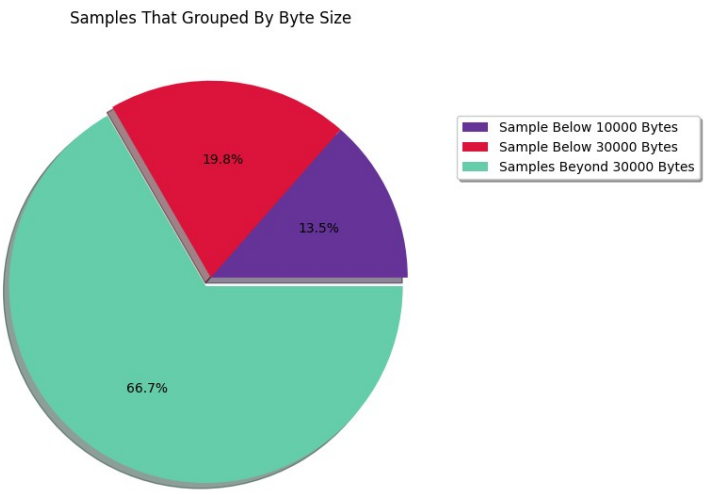
To demonstrate this tool I have gathered a sample of 96 malware of same family and test the tools

```
(malware_classifier) sharmelenubuntu@sharm:~/Desktop/my_project$ python3 main_app.py
Thank you for using this tool to analyse malware
First thing is make sure before continue further you have added Virus Total api key in the api_key.txt
Now you may choose the way you want to analyse your samples:

1. Analyse The Sample Size
2. Analyse Virus Total Match
3. Calculate The Sample Entropy
4. Analyse The Sample Imphash
5. Analyse The Similarity Hash Of The Samples
6. Check The Yara Rules That Matches The Samples
7. Check The Compilation Time

Please Select A Number: 1
File Path of Samples :/home/sharmelenubuntu/Desktop/malware/upatre/upatre/upatre/
Sample name :fd9ea0fd51f5ec8e3b6b35dfbaa59607d60c0fb20e47703cf6bf10b1ec435eec
Byte Size: 10674 bytes
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Sample name :d6c62aacc28e14ab8e4bbd864bef6288a0d477e696ee4482a7bdf859359134a4
Byte Size: 30226 bytes
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Sample name :7230fc080d91aedb0b90e7b425310fa02b8446c31500edc73edd2d863a67b624
Byte Size: 38680 bytes
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Sample name :617606cc7aeb6b8cfd489331125285e87e12be57d340fd00758323129b58f69e
Byte Size: 38792 bytes
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Sample name :e3fa313e52fc9c5961b6d57d439a4134e548d14f08cff31cc9267809db681bec
Byte Size: 29360 bytes
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

From this picture we can see in the console that it takes the directory of the file where all the malware is kept tells its file size and also it generates a graph that have a better visualization of what is the range of the total malware size in that directory



In order to generate this result in the form of pie chart I have used a python package called matplotlib to generate the pie chart

- Virus Total Match (vtmatch.py)

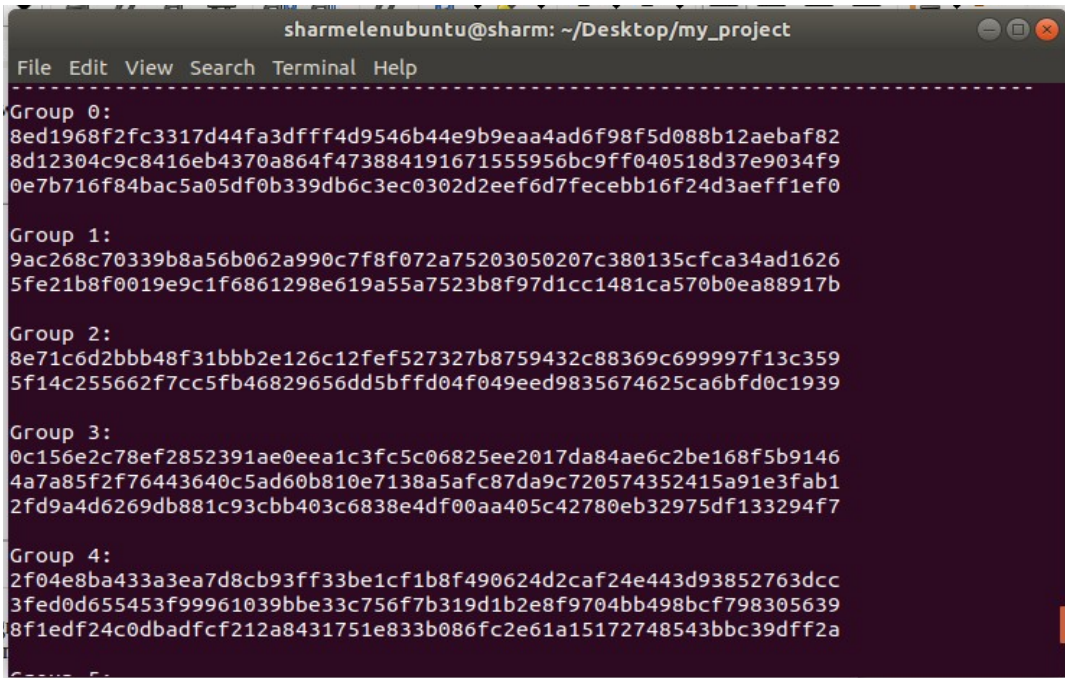
In order to get the total match in virus total for each malware in a file and compile them in bar char to get a meaningful result I have used a python package called virustotal-api that can be used for python coding if we have a api key either premium or public. One of the setback of using public is that we can get the Date of First Seen In the Wild for the scanned malware.

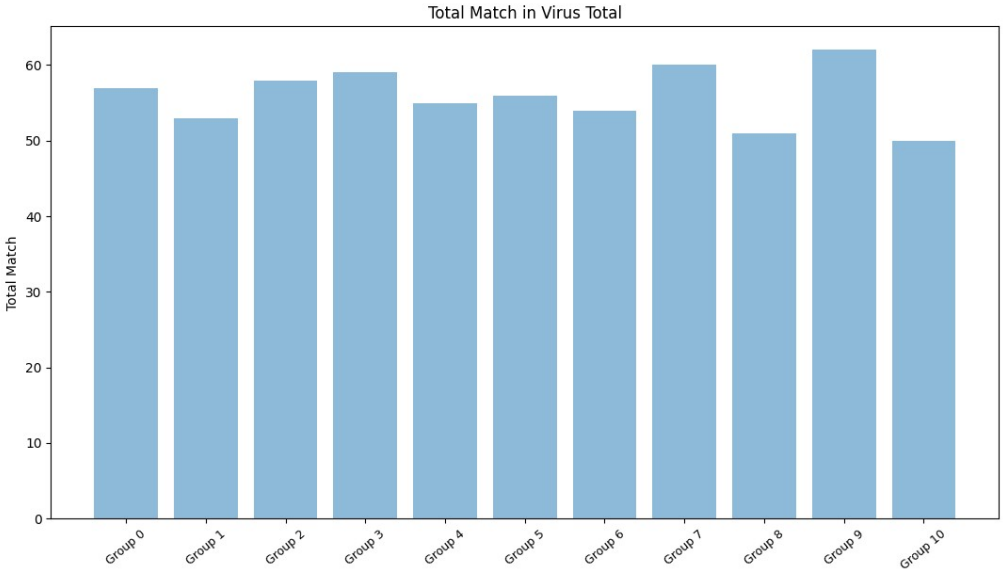
In the code I what I did was basically hash each malware in the file directory and use Virus-Total api to scan the hash and decide if its a potential malware

The setback is that we can only scan 4 malware per minute this is in order not to inflict a huge amount of traffic to the network. So if we have around 100 malwares we have to wait for around 25 minutes to get the result. By the end of the scan result we will get 2 types of bar graph that depicts number of detected malware in the Virus Total and total matches from Top 10 AV companies.

The output result is shown below:

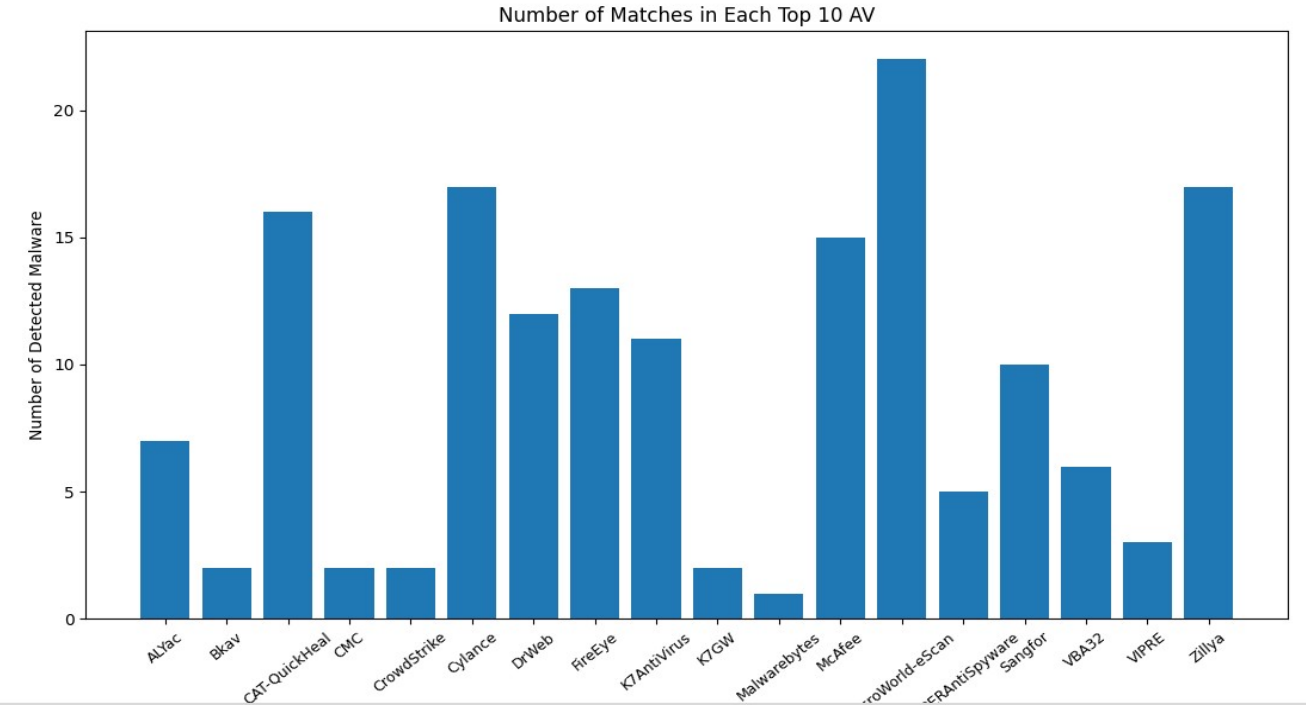
The first graph show the output result of total match of malware in virus total by each group





Each malware are grouped based on how many of them are detected by the virus total we can refer to which group the malware falls in the console. As shown above in the console.

The next graph we will get will show us how many malware is detected by Top 10 AV companies



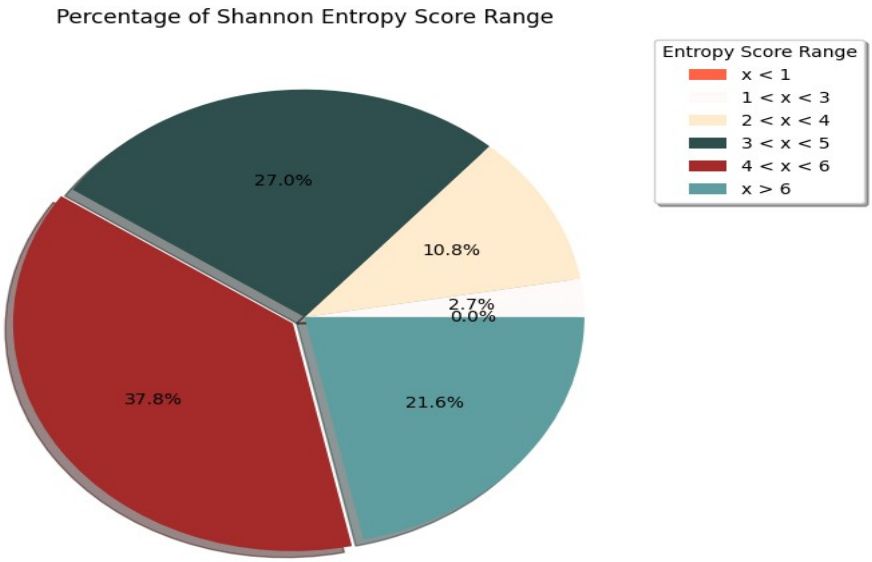
```
File Path of Samples :/home/sharmelenubuntu/Desktop/malware/upatre/upatre/upatre/
Total Number: 69
Detected: 57
Bkav : False
MicroWorld-eScan : True
CMC : False
CAT-QuickHeal : True
McAfee : True
ALYac : True
Cylance : True
Zillya : True
SUPERAntiSpyware : True
Sangfor : True
--------------------------------------------------------------------------------
Total Number: 70
Detected: 53
Bkav : False
DrWeb : True
MicroWorld-eScan : True
FireEye : True
CAT-QuickHeal : False
ALYac : True
Cylance : True
Zillya : True
CrowdStrike : True
Alibaba : False
--------------------------------------------------------------------------------
Total Number: 69
Detected: 57
Bkav : False
MicroWorld-eScan : True
VBA32 : True
FireEye : True
CAT-QuickHeal : True
McAfee : True
Cylance : True
Zillya : True
SUPERAntiSpyware : True
```

To get more details we can see the result of the scan for each malware in the console itself.

- **File Entropy (entropy_calculator.py)**

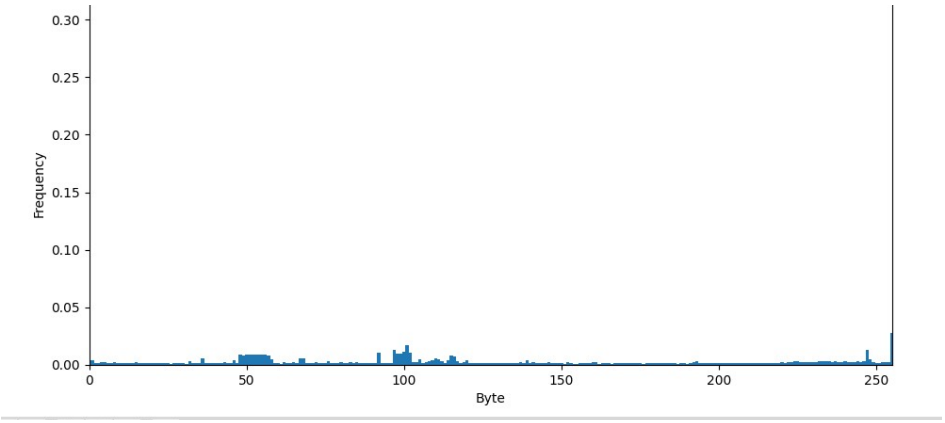
For this aspect I have to create a tool that can be used to calculated the malware's file entropy value which is used to detect if a file is compressed or not. According to Shannnon equation to calculate the entropy of the file if the value of the entropy is closer to 8 the more non orderly the data is and is crtical to security

The tool that I have created can actually scan a multiple scan and create a pie chart by setting a value range and also scan single file to see frequency of byte from 0 to 255 bytes



This pie chart shows group of malwares that has been seperated in an entropy score range value

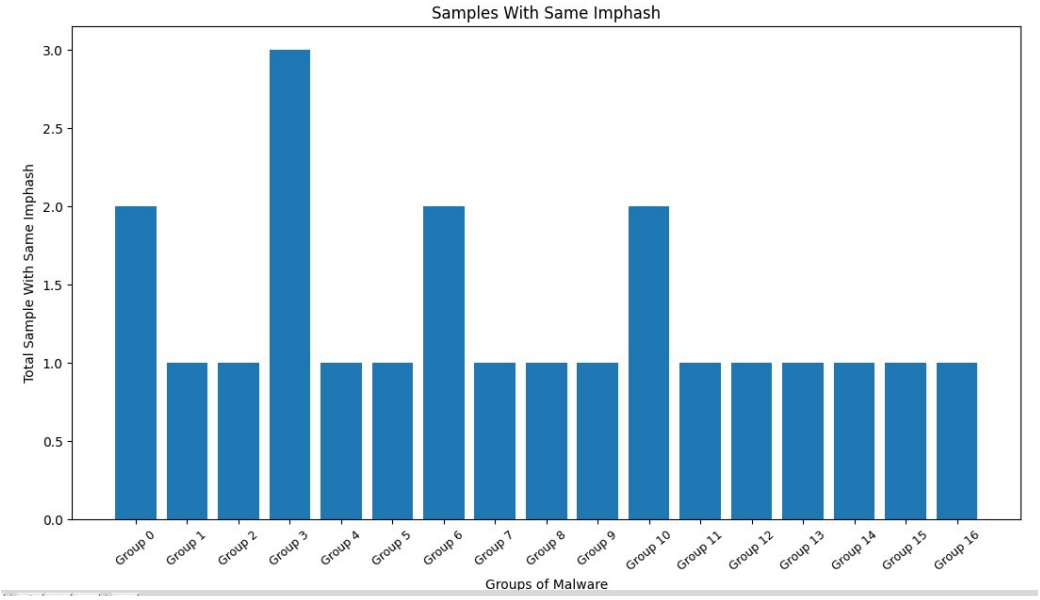
Other than that we can also choose a single malware to see the frequency of bytes from 0 to 255 to get more detailed view and how random the bytes of the sample



- Analyse The Imphash of Malwares (imphash.py)

For this I have actually used pefile package in python that can be used to read executable file and retrieve the import module from each of the malware and hash them. Once hashed I compare each one of them and categorize them based on their imphash similarity. The hash that I used was md5.

In the sample output we can see the similar imphash for the malwares



```
Please Select A Number: 4
File Path of Samples :/home/sharmelenubuntu/Desktop/malware/upatre/upatre/upatre/
Group 0
Imphash Value: eb1878730a247c371ddbdaa8b40c392b
8ed1968f2fc3317d44fa3dfff4d9546b44e9b9eaa4ad6f98f5d088b12aebaf82
1dc30af6b71627940b9c2357b41e22cc39afc6f7d46faca35238fa21adfa2c3a
##########################################################################
Group 1
Imphash Value: 3011ff1499499c2018225149820962bc
9ac268c70339b8a56b062a990c7f8f072a75203050207c380135cfca34ad1626
##########################################################################
Group 2
Imphash Value: 0d57379f90822c7809fe2642876db4da
8d12304c9c8416eb4370a864f4738841916715559 56bc9ff040518d37e9034f9
##########################################################################
Group 3
Imphash Value: f5ba129d6a0880ed89f2f9583985bd00
0e7b716f84bac5a05df0b339db6c3ec0302d2eef6d7fecebb16f24d3aeff1ef0
1d61db72c7b2b8af43757eaffe82231b2fa830907d04dabcd96c395fb1c91244
4b3dc7d9aa26ffd6115982da67858ab2d7a1f576de1ed3b408615e3a876f8a5c
##########################################################################
Group 4
Imphash Value: df33e23912a96fd6550db82fc68815c6
8e71c6d2bbb48f31bbb2e126c12fef527327b8759432c88369c699997f13c359
##########################################################################
Group 5
Imphash Value: c03467f55d9bb6f97e807ada405f53d7
0c156e2c78ef2852391ae0eea1c3fc5c06825ee2017da84ae6c2be168f5b9146
##########################################################################
Group 6
Imphash Value: 9e27316924bf6c5e4f9638ab4e71575c
4a7a85f2f76443640c5ad60b810e7138a5afc87da9c720574352415a91e3fab1
5e39e1b3daaf68a1a62a1ac9cf53466e53718d50eac088811559d1dd4736e684
##########################################################################
Group 7
Imphash Value: 5a2e47555343d04fd86834925ddfd386
2f04e8ba433a3ea7d8cb93ff33be1cf1b8f490624d2caf24e443d93852763dcc
##########################################################################
```

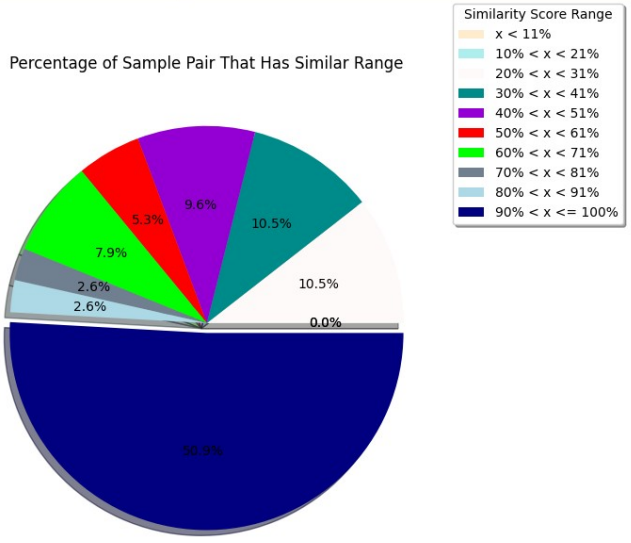
So basically what this tools does is that it groups each malware that hash the same imphash and count them we can refer to the groups of malware in the console.

- Analyse The Similarity Hashes of Each Malware Pairs (similar_hash.py)

For this aspect I have considered using to hash technique namely SSDEEP and TLSH that is used to compare the malware sample of their similarities with each other.

First we will talk about about SSDEEP. So basically what SSDEEP is that compares the hashes of each malware pair and tell us their similarity percentage. So basically what I did was installed a tool called SSDEEP in my linux and run the SSDEEP tool in my malware directory to get the results. Once I have obtained the result I wrote a python script to tabulate the result in a more visually appealing form.

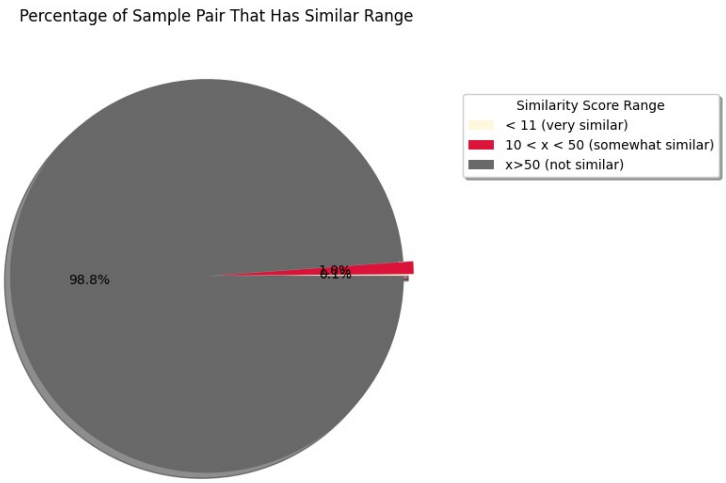
This is the sample result that we get based on the score range of the malwares that I scanned



Next we will look in TLSH so

basically for TLSH I used a python package called python-tlsh that can be used to compare the hashes for each file

This is the sample result that we can get from TLSH hash

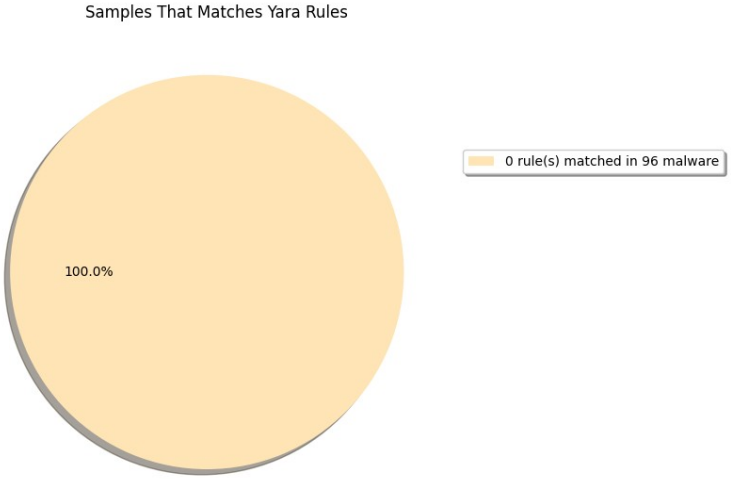


Unlike SSDEEP for TLSH the lower the score of a malware sample pair the more similar they are.

- Yara Matching Rule For Malware (yara_rules.py)

To create this tool I have used a python package called yara-python that is used to compile yara rules and use it to compare any matching rules with the scanned malware. For this tool what I did was create a few rules with yara and store it in a folder and let script a python code to scan and compile the yara rules. Once the rules has been compiled I used the compiled rules to scan the malware directory to scan for any matching rule.

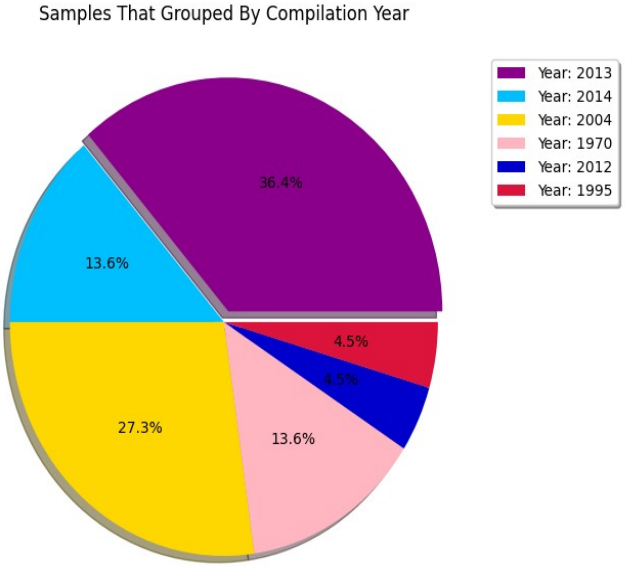
For example this is the result I get when I scanned my malware directory



- Compilation Time(compilation_time.py)

For this aspect I have used a python package called pefile to actually get the compilation time of each malware and create a pie chart that based on the compilation year. So the pefile will do is that it will read the module of an executable file.

We can see the example of the output of this tool below:



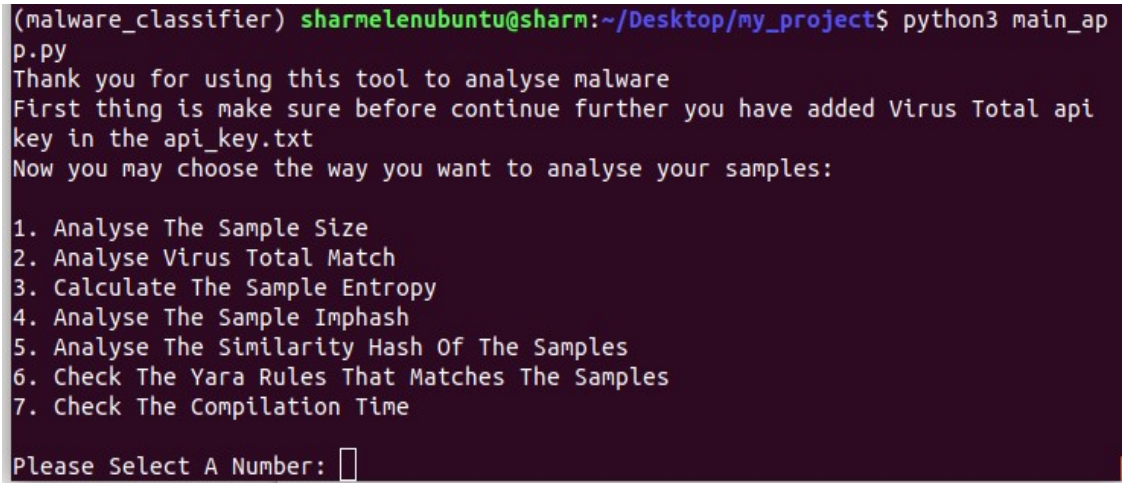

```
------------------------------------------------------------
Year: 2013
------------------------------------------------------------
Sample: 8ed1968f2fc3317d44fa3dfff4d9546b44e9b9eaa4ad6f98f5d088b12aebaf82
Sample: 8e71c6d2bbb48f31bbb2e126c12fef527327b8759432c88369c699997f13c359
Sample: 5e39e1b3daaf68a1a62a1ac9cf53466e53718d50eac088811559d1dd4736e684
Sample: 1dc30af6b71627940b9c2357b41e22cc39afc6f7d46faca35238fa21adfa2c3a
Sample: 5fe21b8f0019e9c1f6861298e619a55a7523b8f97d1cc1481ca570b0ea88917b
Sample: 4a7ad608b5dc76ada03802e2964525bacae4dbf35ef328e5846986032445fb9c
Sample: 2e23478781a7edf55cdc2055684aee13b30e9ea7059dbb97ad51b23f0ad91427
Sample: 2fd9a4d6269db881c93cbb403c6838e4df00aa405c42780eb32975df133294f7
------------------------------------------------------------
Year: 2014
------------------------------------------------------------
Sample: 9ac268c70339b8a56b062a990c7f8f072a75203050207c380135cfca34ad1626
Sample: 0c156e2c78ef2852391ae0eea1c3fc5c06825ee2017da84ae6c2be168f5b9146
Sample: 2f04e8ba433a3ea7d8cb93ff33be1cf1b8f490624d2caf24e443d93852763dcc
------------------------------------------------------------
Year: 2004
------------------------------------------------------------
Sample: 8d12304c9c8416eb4370a864f4738841916715559 56bc9ff040518d37e9034f9
Sample: 3fe1d0885030ab7385fb3ff66160bbcf89ca0fe73a9cc02bbd25e0f6aaae6124
Sample: 5f14c255662f7cc5fb46829656dd5bffd04f049eed9835674625ca6bfd0c1939
Sample: 8f1edf24c0dbadfcf212a8431751e833b086fc2e61a15172748543bbc39dff2a
Sample: 07f2e52a903baefdd6615c95505f04533a9f51ebf3533f34a8ca8d539ca7be4c
Sample: 7c9f3aa70a4e6387da0c3e5a113cd77eaa41b1d29e02ba206e8bc9040e38c7ed
------------------------------------------------------------
Year: 1970
------------------------------------------------------------
Sample: 0e7b716f84bac5a05df0b339db6c3ec0302d2eef6d7fecebb16f24d3aeff1ef0
Sample: 1d61db72c7b2b8af43757eaffe82231b2fa830907d04dabcd96c395fb1c91244
Sample: 4b3dc7d9aa26ffd6115982da67858ab2d7a1f576de1ed3b408615e3a876f8a5c
------------------------------------------------------------
Year: 2012
------------------------------------------------------------
Sample: 4a7a85f2f76443640c5ad60b810e7138a5afc87da9c720574352415a91e3fab1
------------------------------------------------------------
```

We can refer which compilation year the malware falls in the console to get more information.

- The main script that is used to access all the tools that was mentioned above (main_app.py)

This tool is basically used as way to access all the tools that was mentioned above running this script it will give us a number of option that we can choose that comprise of all the tools that I have mentioned above. All we have to do is just select the number of the tool that we interested in and give the required input.

The picture below shows the look when we run the main_app.py script



This tool can be downloaded at this link https://github.com/Sharmelen/Malware_Analyzer.git at this repository please do read the instructions on how to use this tool so that no problems will be arise or face any crashes.