**1. Problem Statement**

For my project, I intended to modify upon an existing VirusTotal scanning tool. The original tool would submit the hash for a given file to VirusTotal, and return data like detection rates, associated malware, relevant PCAP files, and generated JSON reports. However, this tool would only accomplish this for individual files, requiring the user to repeatedly resubmit new files if multiple files needed to be scanned. Other tools require that the list of files be placed into a text file first before submission. I modified the original tool by changing it to take a directory as user input instead of single files, allowing for the return of reports for all the files within said directory with one command. I attempted to achieve this by using python to create the loop for finding all files within the given directory, and finding the absolute path names for those files.

**2. Research Performed**

**To determine the existing work already available in this area I looked at the following resources:**

**2.1 Articles**

https://digital-forensics.sans.org/blog/2015/03/19/just-in-time-virustotal-hash-checking/

https://www.virustotal.com/en/documentation/public-api/#scanning-files

**2.2 Tools/Code Base**

https://github.com/Xen0ph0n/VirusTotal\_API\_Tool

https://docs.python.org/3/howto/argparse.html

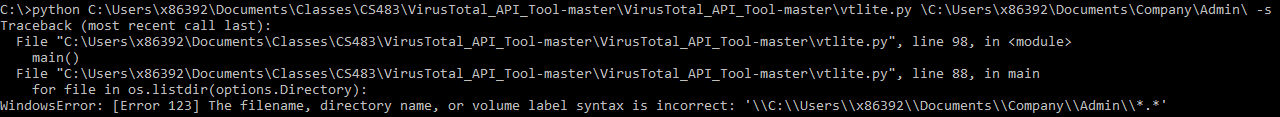
http://www.diveintopython.net/functional\_programming/finding\_the\_path.html

https://docs.python.org/2/library/os.html

**3. Assumptions and Design Decisions**

In order to achieve my goals, I assumed that the user was short for time and desired an expedient way to classify numerous files by their risk and begin triage. Given this, the user would be unwilling to submit each potential file to VirusTotal one at a time. I also assumed that the directory housing all suspicious files contained no further sub-directories. It must also be assumed that the user has signed up on VirusTotal and possesses the free API key which can be found under their profile settings.

**4. Status of the Project**

I was unable to achieve all of the goals of this portion of the project. Modifications in the code returned errors when sub-functions within the python script used for MD5 called on the pathnames for the files. I believe the source of the error lies in the Directory argument option. A Windows error is also returned based on the syntax of the given suspected directory. An example of attempting to scan a directory named C:\Users\x86392\Documents\Company\Admin generated the following errors. 

**5. Program Execution Instructions** (This should tell me how to run your program, how to interpret the results, and possibly provide some screenshots)

Open the zergRush.py file, and on Line 13, paste in your API key. Save and exit.

To execute the program on Windows, open Command Prompt:

If you already have Python included under your Path environment variable:

> python ../../zergRush.py ../../mySuspectedDirectory –s

If Python is not included under the Path environment variable, it can most likely be found

within the following directory : C:\Python27\

> C:\Python27\python ../../zergRush.py ../../mySuspectedDirectory –s