**CIS-549 TERM PROJECT**

**Python Based Anti-Virus Application**

**Team 2**

# Table of Contents

[**Table of Contents**](#_heading=h.pwdch8krbrcm) **2**

[Abstract](#_heading=h.1052j9ege3uh) **3**

[Introduction](#_heading=h.9eg6k2hjs141) **3**

[Related/Prior Work](#_heading=h.jijv30su3wv3) **3**

[Experimental Setup](#_heading=h.88v62z6rbp3e) **4**

[Technical Information](#_heading=h.kawgavujibqd) **4**

[Data Results](#_heading=h.2wqh27egabjy) **4**

[Data Collection and Analysis](#_heading=h.n291acel1ja6) **10**

[Remaining/Future Work](#_heading=h.80hmp0g6ggol) **11**

[Team Member Contributions](#_heading=h.139qacburi4d) **11**

# Abstract

With an increasing amount of new and exciting technologies being introduced into the marketplace including computers, IoT devices, and in-vehicle networking, the risk for additional virus attack surfaces such as ransomware, malware, and buffer overflow attacks also increases. Today we define a computer virus as an executable program that can multiply itself quickly on multiple machines and possibly over a network that can cause malicious havoc such as denial of service attacks, identify theft, the search for personal or financial information and many more. The impacts of ransomware and malware attacks can halt an organization’s business and cause major financial disasters, which makes protecting technological assets essential. Standard anti-virus programs (e.g., Norton, McAfee, etc.) may not correctly identify all threats which may have potential to cause harm. Therefore, our project looks at developing a Python-based antivirus application utilizing the VirusTotal API. VirusTotal is an API that allows any user to upload files that they feel could be potentially dangerous. Once a file is uploaded, it looks at a variety of anti-virus programs at a very low level. The main focus on our application will be to upload files from datasets that have different kinds of extensions, \*.apk, \*.dex, and \*.oat files. After uploading, these files will be scanned and analyzed to produce a single JSON output to display stats and analytics such as virus type, total viruses detected, and accuracy of the virus scan. In the following paragraphs, we will describe in more detail any related work that has been done, the technology and coding we used to build our application, and the overall analysis.

# Introduction

Ransomware is an attack based on an adversary breaking into a machine and encrypting various important files that could have a high or extreme impact on the individual or organization. In order for these important files to be released the individual or organization has to pay a certain amount of money demanded by the adversary for them to be released or unencrypted. According to Bloomberg, “The average payment in 2020 was $312,493, according to Palo Alto Networks, a 171% increase over the previous year” (Turton, 2021). How do we know that our files are safe from potential ransomware attacks? Since most viruses are designed to run in the computer's background, users often do not know when they are infected or infecting others. In order to try and avoid this, we install and run local antivirus programs that are designed to scan, detect, and remove these viruses. However, what if the local antivirus program doesn’t detect a virus at a low enough level because it’s designed to run in memory, or by hashing.

Therefore, the idea of this project is to develop an antivirus application that can scan and detect with more precision using the VirusTotal. With this project, we demonstrated how multiple local files with \*.apk, \*.oat, and \*.dex files can be uploaded simultaneously. With VirusTotal we obtain results quickly and efficiently. MD5 (message-digest Algorithm) is a hashing algorithm used to accept messages of any length as input and return a digest value of a fixed length that can be used to authenticate the original message. Additionally, there are API (Application Programming Interface) keys that allow us to access the internal features of the program. Streamlit was used as our frontend api for a smooth user interface.

# Related/Prior Work

VirusTotal is open source and many projects have been developed to upload files to the VirusTotal website. We reviewed two projects, and the first one is called, “checkVT” (Almonte, 2021), which uses URLs to scan and verify with VirusTotal. The second project was an application called “VirusTotal Uploader” (Tulach, 2021), that is a simple API that you drag a file to and have VirusTotal perform the scans, detections, etc. In order to use these applications, you have to know how to do some backend coding as well as being savvy with command lines or is just a simple drop/drag box. Both of these applications provide no analytics or an easy-to-use web interface for selecting a file or multiple files..

Therefore, we’ve built an application that combines easy to view analytics, stats, and information about the file or files uploaded to VirusTotal. We allow users to upload multiple files in order to detect ransomware and VirusTotal will provide information about the virus types, the amount of virus detected, accuracy of the scan, and number of antiviruses that did not time out. Hash values and API keys are used to differentiate the files uploaded to authenticate input files from the terminal. The output displays the type of virus in detail along with its hash value.

# Experimental Setup

Streamlit is used to communicate between the local machine and the virus total website. The dataset contains different kinds of files such as oat, apk and, dex files. Users need to install libraries such as streamlit, pandas, numpy and requests in order to run this application (command to install: pip install streamlit pandas requests ). After installing all the libraries that are required, the program is executed (command to execute: streamlit run <filename>.py). The files are then uploaded into a webpage. Hash values were generated for each and every file which is unique. It scans and produces a JSON format output containing stats and analytics such as virus type, total viruses detected and accuracy of the scanning. Along with the results graphs are produced for every section.

# Technical Information/Architecture

**Software used:** Visual Studio Code, Linux 20.04.

**Programming Language:** Python.

**Datasets:** .apk, .dex,and .oat dataset

# Data Collection, Results, and Analysis

The output is in the form of a single \*.json file that shows the name of each segment along with the name of the virus it has the most. This is followed by its hash value with its file type. Furthermore, it shows the total virus detected, the number of anti viruses that didn't time out, and the accuracy detection percentage.

Here, the file is uploaded to the program and is scanned for viruses. After submitting the file, a \*.json file is created.

def send\_file(file):

url = 'https://www.virustotal.com/vtapi/v2/file/scan'

params={'apikey': 'd5166c48ffb2e3a2a43ddc9dcf0c5d3f2256e4d77a263b466a888ac3337e1ce7'}

try:

file\_path = file\_loc1+r"\\"+file.name

files = {'file': (file.name, open(file\_path, 'rb'))}

except:

try:

file\_path = file\_loc2+r"\\"+file.name

files = {'file': (file.name, open(file\_path, 'rb'))}

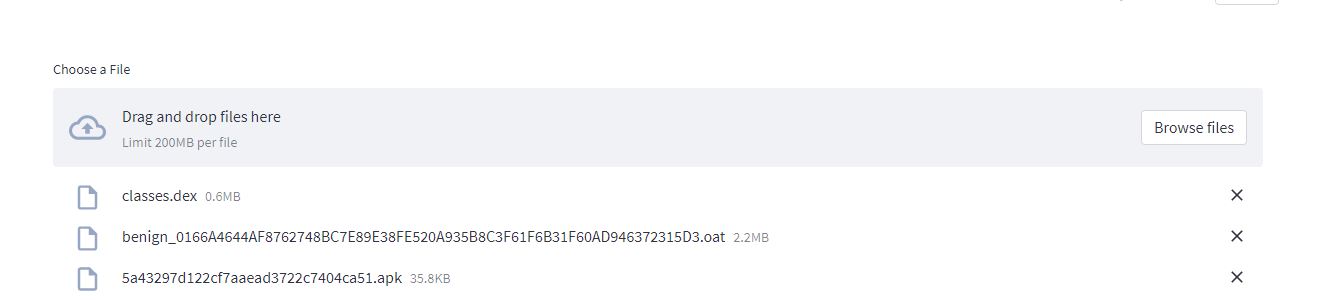
except:

files = {'file': (file.name, open(file.name, 'rb'))}

response = requests.post(url, files=files, params=params)

return get\_report(response.json())

In this case, we uploaded all three Datasets and submitted them to VirusTotal for analysis.

****

This is the JSON file that is created, which contains every detail about the file, such as if the file is infected, Anti Virus, and accuracy.

def format\_report(data):

total\_virus\_outputs = {}

output\_dict = {}

try:

file\_type = data['data']['attributes']['meaningful\_name'].split('.')[1]

except:

file\_type = data['data']['attributes']['type\_description']

detect\_count = data['data']['attributes']['last\_analysis\_stats']['suspicious'] + \

data['data']['attributes']['last\_analysis\_stats']['malicious']

total\_av=sum(data['data']['attributes']['last\_analysis\_stats'].values())-data['data']['attributes']['last\_analysis\_stats']['timeout']

accuracy = detect\_count/total\_av

total\_virus\_outputs['detect\_count'] = detect\_count

total\_virus\_outputs['total\_av'] = total\_av

total\_virus\_outputs['accuracy'] = accuracy

for k, v in data['data']['attributes']['last\_analysis\_results'].items():

if v['category'] == 'undetected' or v['category'] == 'type-unsupported':

total\_virus\_outputs[k] = 0

elif v['category'] == 'timeout':

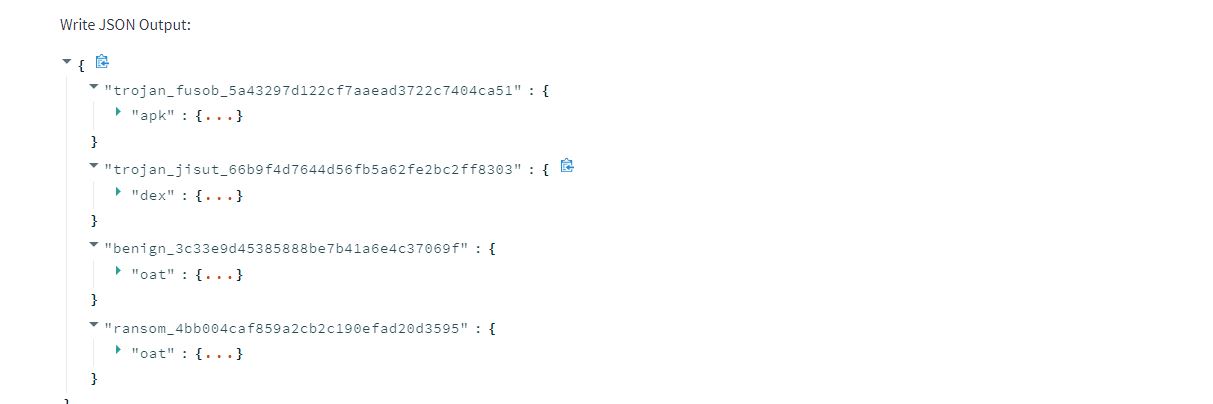
pass

else:

total\_virus\_outputs[k] = 1

output\_dict[file\_type] = total\_virus\_outputs

return output\_dict



We will get the results in this file as total viruses detected and total antivirus count and percentage accuracy.

list\_of\_names=[]

detect=[]

total\_av=[]

accuracy=[]

for k,v in aggr\_report.items():

list\_of\_names.append(k)

for k2,v2 in v.items():

detect.append(v2['detect\_count'])

for k3,v3 in v.items():

total\_av.append(v3['total\_av'])

for k4,v4 in v.items():

accuracy.append(v4['accuracy'])

st.write("Write File Names/Count: ", list\_of\_names)

st.write("Write Total Detected: ", detect)

st.write("Write Total AV Count: ", total\_av)

st.write("Write Accuracy Percentage : ", accuracy)

****

The below graph depicts the total number of viruses detected.

def load\_detect\_data():

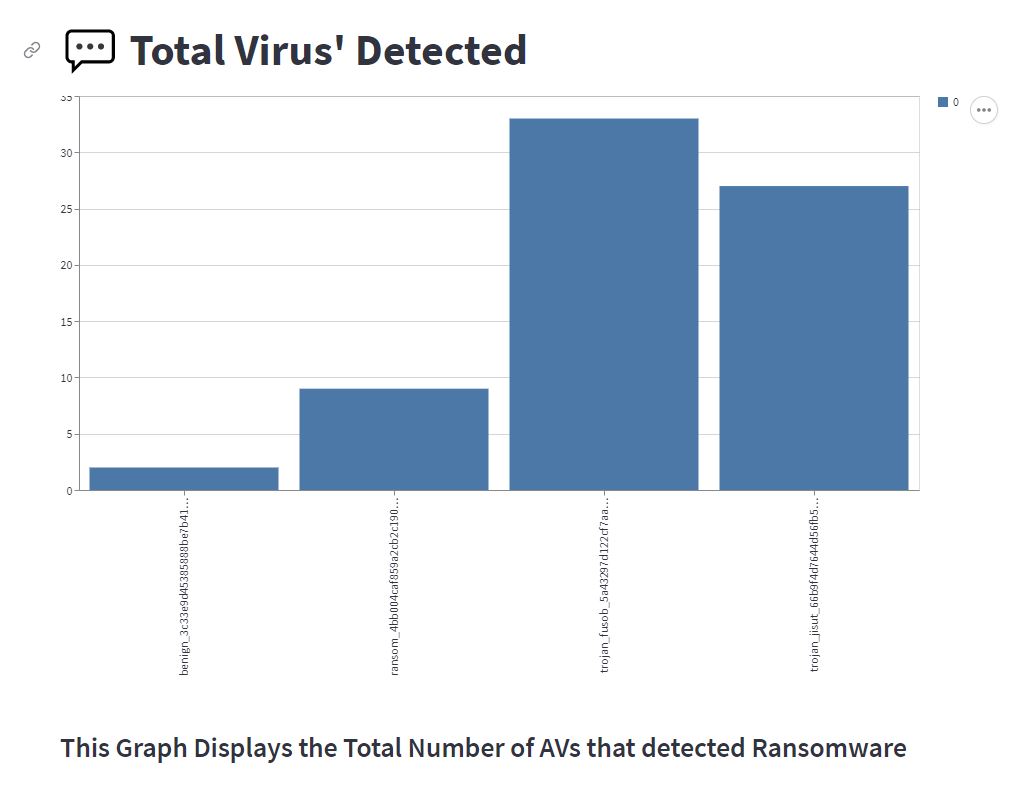
st.title(" Total Virus' Detected")

df = pd.DataFrame(detect, list\_of\_names)

st.bar\_chart(df, width=900, height=900, use\_container\_width=False)

st.subheader("This Graph Displays the Total Number of AVs that detected Ransomware")

chart\_data = load\_detect\_data()

****

The below Graph shows the Amount of Antivirus detected.

def load\_total\_av\_data():

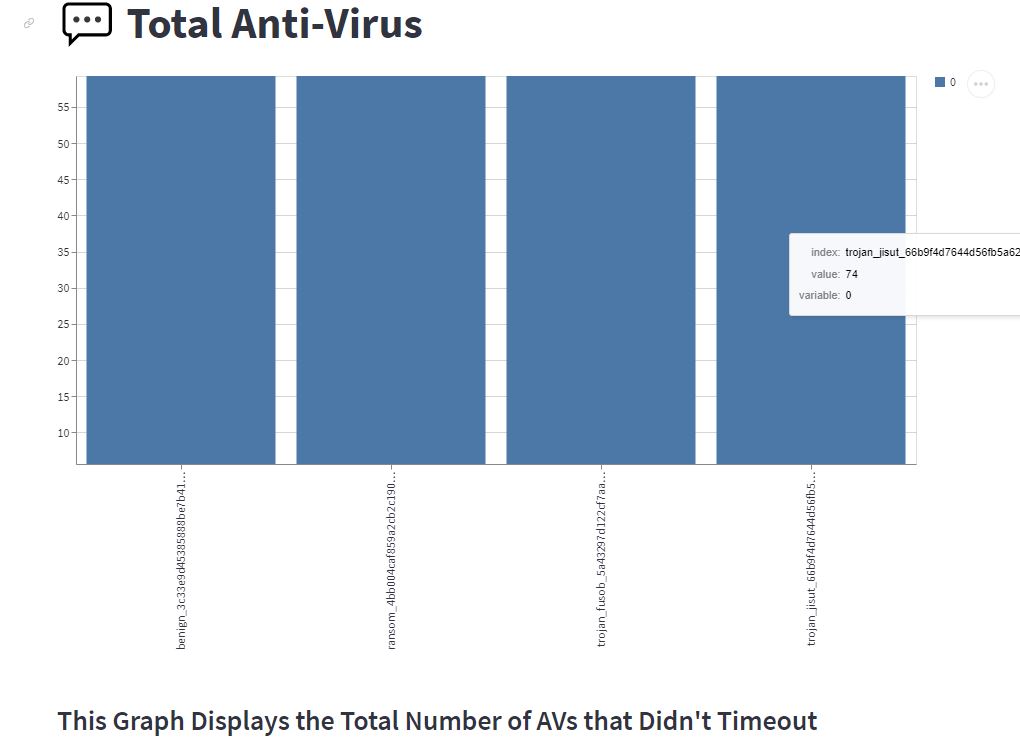
st.title(" Total Anti-Virus")

df = pd.DataFrame(total\_av, list\_of\_names)

st.bar\_chart(df, width=900, height=600, use\_container\_width=False)

st.subheader("This Graph Displays the Total Number of AVs that didn't timeout")

chart\_data = load\_total\_av\_data()

****

The below graph shows the accuracy percentage of the AntiVirus Program.

def load\_accuracy\_data():

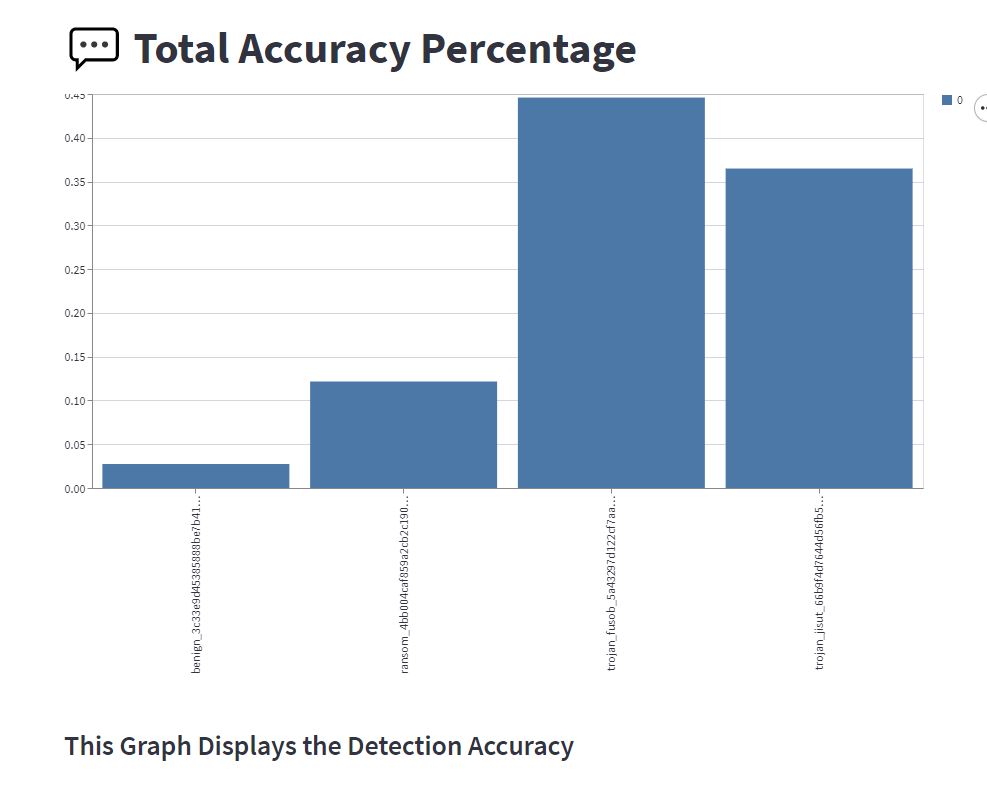
st.title(" Total Accuracy Percentage")

df = pd.DataFrame(accuracy, list\_of\_names)

st.bar\_chart(df, width=900, height=600, use\_container\_width=False)

st.subheader("This Graph Displays the Detection Accuracy")

chart\_data = load\_accuracy\_data()

****

# Remaining/Future Work

There currently is no remaining work left to perform for this project. However, the application is designed for manual intervention where an individual has to manually add files to our program that uploads the files to VirusTotal to be scanned and analyzed. An improvement to our application or future work, would have the application auto analyze the computer/file system daily and give a report on any malicious files. This is advantageous because it will encompass the entire system and remove the human aspect of the virus finder and add system-wide security. Likewise, we can add some additional charts, stats, and have the program automatically delete/contain files that are found dangerous.

# Team Member Contributions

Each team member had some contributions to developing and building the application as well to the analytics and reporting. Below is a breakdown of the contributions.

* Application development completed by Frank and Jason with assistance from Fathi
* Reporting and analytics completed by Frank and Jason with assistance from Hareesh
* Team meeting discussions, brainstorming, and represented by each team member
* Ad-hoc discussions/meetings via Discord was represented by each team member
* Demonstration completed by each team member