****

**Application Security**

**Continuous Assessment 3**

**VirusTotal Report Parser**

***Sean Guthrie***

***B00073216***

Department of Informatics

School of Informatics and Engineering

Technological University of Dublin,

Blanchardstown Campus

Dublin 15.

**BN311 Bachelor of Science in Computing**

**Digital Forensics & Cyber Security**

**Application Security**

**05/01/2020**

# Plagiarism Declaration

**TECHNOLOGICAL UNIVERSITY OF DUBLIN**

**DEPARTMENT OF INFORMATICS**

**HIGHER CERTIFICATE IN SCIENCE IN DIGITAL FORENSICS AND CYBER SECURITY**

**LECTURER STEPHEN O’SHAUGHNESSY**

**DECLARATION ON PLAGIARISM**

I declare that the work I/We am(are) submitting for assessment by the Institute examiner(s) is entirely my(our) own work, except where the author or source has been duly referenced and attributed.

I confirm that this material has not been previously submitted for a degree or any other qualification at TUDublin or any other institution. I further confirm that I have read and understood the Institute policy on plagiarism in assignments and examinations (3AS08.doc) and that I am not, so far as I am aware, in breach of any of these regulations.

Signed: Sean Guthrie

Date: 05/01/2020

Contents

[Plagiarism Declaration 2](#_Toc29591736)

[Abstract 4](#_Toc29591737)

[Introduction 4](#_Toc29591738)

[Libraries Used 4](#_Toc29591739)

[Application Walkthrough 5](#_Toc29591740)

[Upload File 6](#_Toc29591741)

[Get File Report 9](#_Toc29591742)

[Malicious File Example 10](#_Toc29591743)

[Upload URL 12](#_Toc29591744)

[Get URL Report 15](#_Toc29591745)

[Malicious URL Example 16](#_Toc29591746)

[Malware Hash 17](#_Toc29591747)

[Malicious Malware Hash Example 19](#_Toc29591748)

[Code Walkthrough 20](#_Toc29591749)

[Main Menu Function 20](#_Toc29591750)

[Upload File Function 22](#_Toc29591751)

[Save File Function 23](#_Toc29591752)

[Read File Function 23](#_Toc29591753)

[Scan URL Function 25](#_Toc29591754)

[URL Report Function 26](#_Toc29591755)

[Malware Hash Function 27](#_Toc29591756)

[Unresolved bugs 30](#_Toc29591757)

[Conclusions 30](#_Toc29591758)

[References 30](#_Toc29591759)

# Abstract

This report will outline the workings of the VIrustotal Parser application, a GUI based Python application that can send files, URL and hashes to VirusTotal via an API, and reproduce reports received from VirusTotal.

# Introduction

This report details the workings of the Virustotal Parser application, which came bundled with this report. This application is entirely programmed in Python 3.6. This application takes a file, a URL or a Hash of a malware and compares it to the VirusTotal database, using an API. The API then sends back a report, which the application presents to the user. The user is also able to save these reports to their machine in order to inspect at a later date. This application is GUI based to provide easier accessibility to the user.

## Libraries Used

For this application, it is recommended to have the following Libraries installed:

Virustotal\_python – API for Virustotal

* Installation Command: pip install virustotal-python

Tkinter – GUI Programming for python

* Already preinstalled, in the scenario of it not being available:

Linux: sudo apt-get install python-tk

PIL – Hosts Image on GUI

* Installation Command: pip install Pillow

Hashlib – Generates hash from the file

* Already preinstalled

JSON – allows JSON files to be created/read/wrote

* Already preinstalled

# Application Walkthrough

For this section of the report, we will be providing a walkthrough of the use of the application, in relation to how an end user would use it. Due to this application being GUI based, the walkthrough will primarily show the step by step processes with the GUI. We will also be providing examples of this application being used as to show the effectiveness of the application.

We will start this with the menu page of the application, which has several buttons, allowing the user to select an option from the menu.

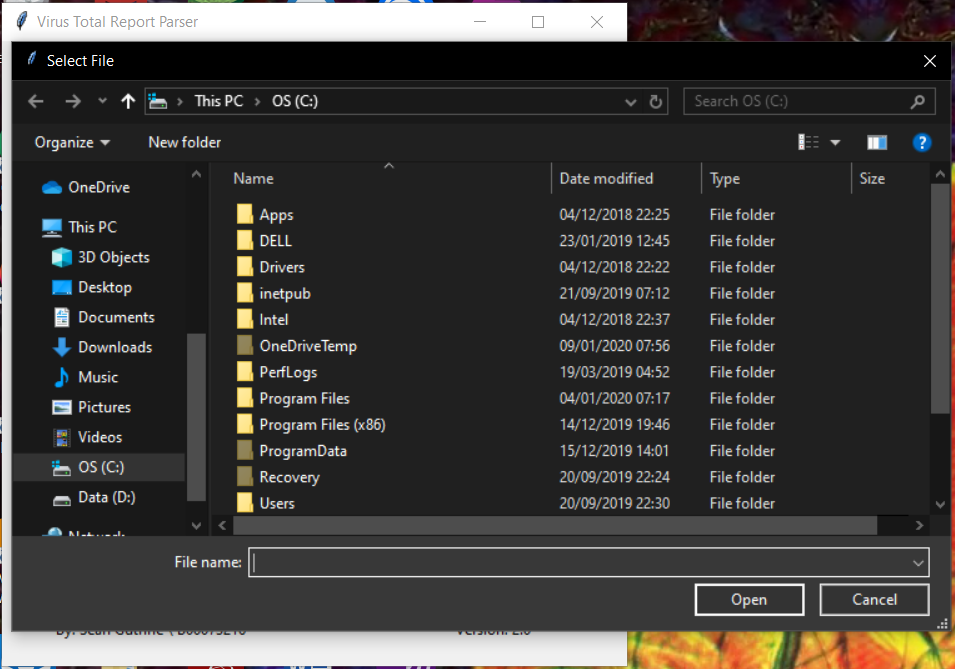


The options that are available are:

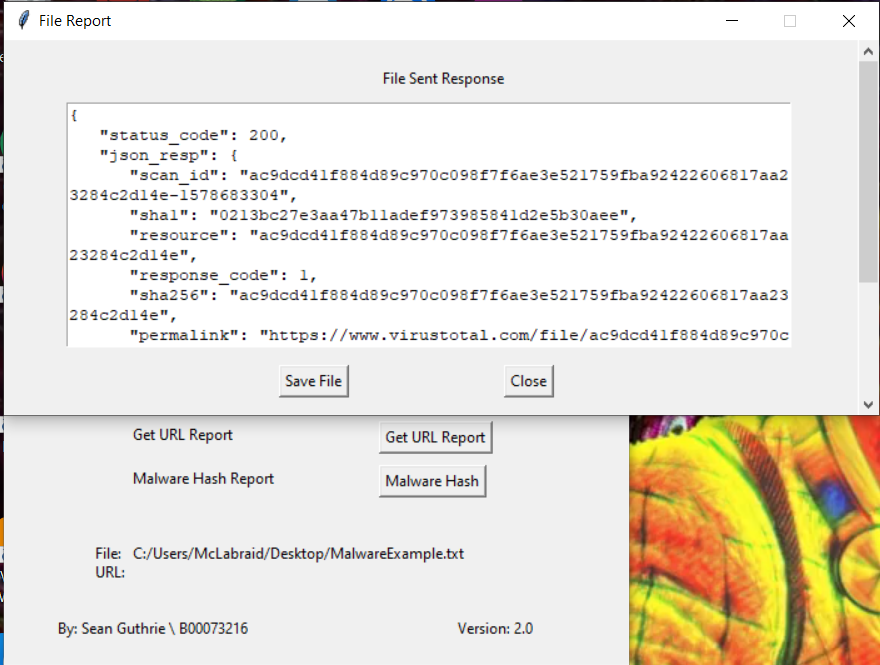
* Upload a file: A window is opened for the user to select an file to be analysed
* Get file Report: get the report for the file
* Upload URL: Input a URL to be inspected
* Get URL Report: Get report for the submitted URL
* Malware Hash Report: Provide a SHA256 of a malware, and a report is returned.

### Upload File

When the “Upload a File” option is selected, we are prompted with a “select file” window, from here we can select a file to upload to the software.



Upon selecting the file, we are given a window with the response from the application upon the submission of the file, we have the choice to save this report or to close the window.

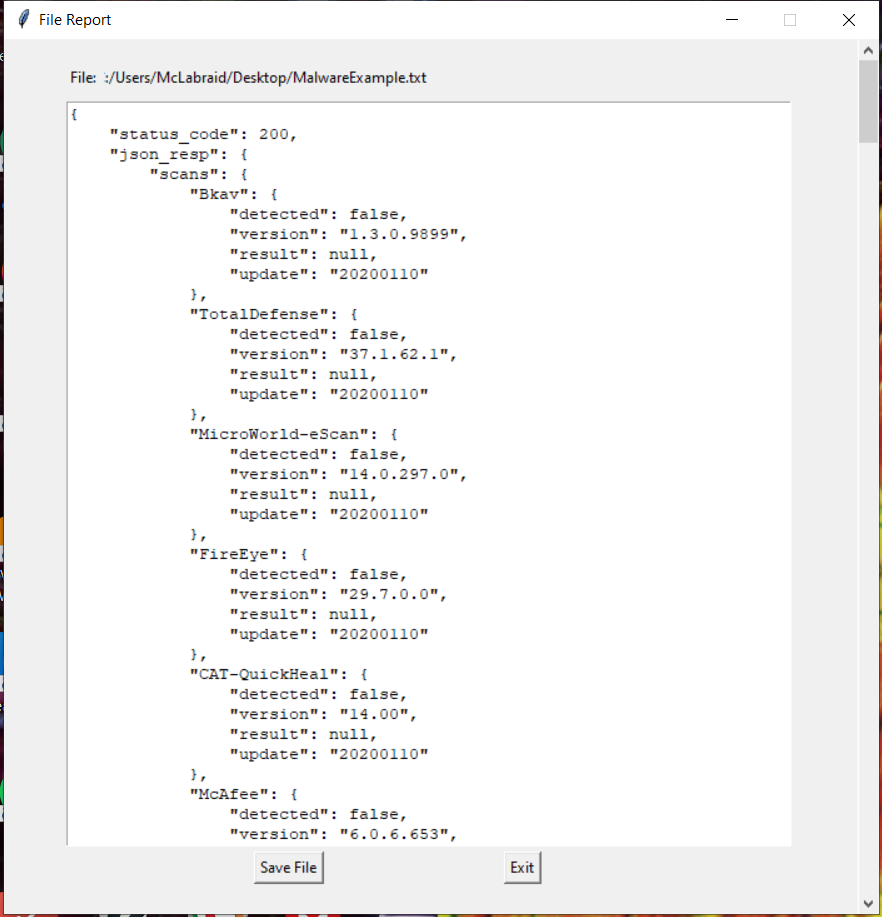


After this, we are returned to the main menu, with now the file location appearing on the main menu as a reminder of it being selected.

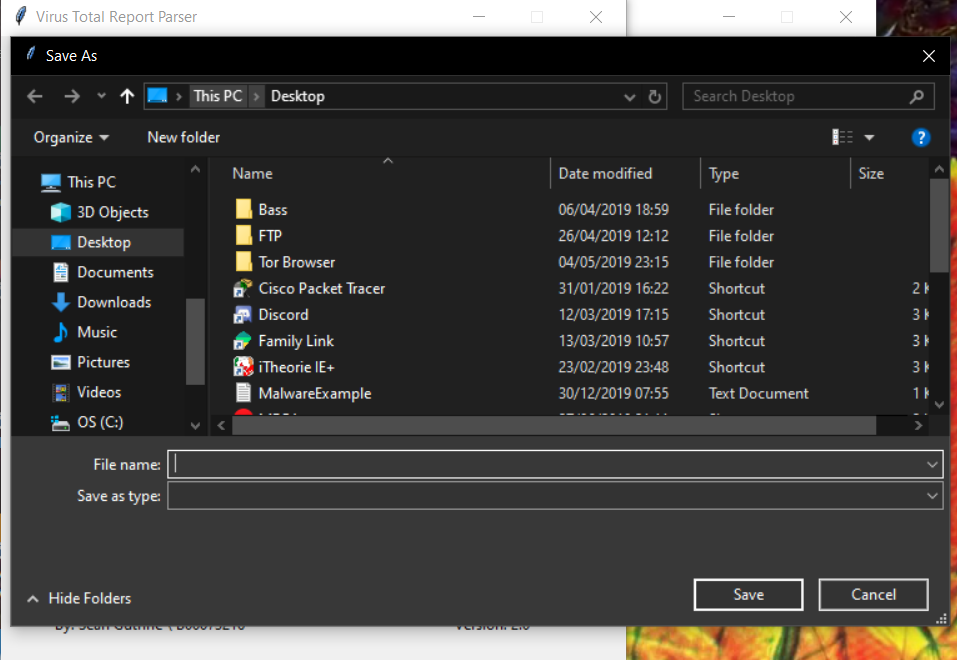


### Get File Report

From here, we select the next option “Get File Report”, clicking this option will provide a report of the aforementioned file that was uploaded. This is done by getting the SHA256 of the file and sending it to the virustotal API to compare and return the report.



From here we are able to save the report or close the window. Choosing to save will prompt the user with a save window.

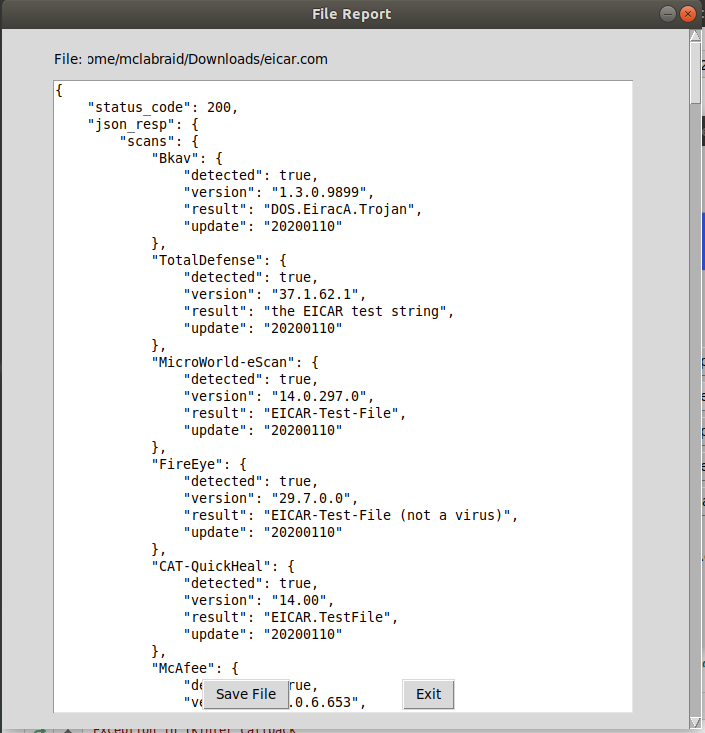


After saving and closing the new window, we are back at the main menu.

### Malicious File Example

To show the capabilities of this feature, we used the fake EICAR-Test Virus to check it’s functionality, we uploaded the file to the application, then got a report that it does contain a “Virus”.

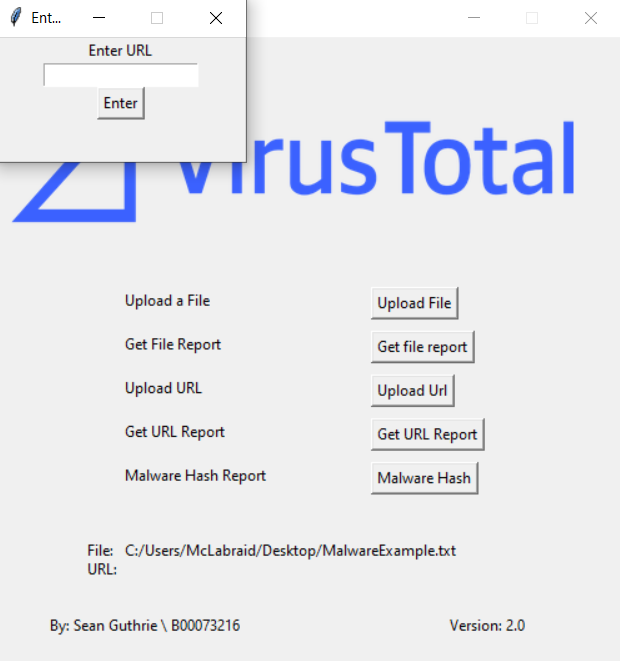
Note: This section was done on a Ubuntu Linux machine due to my Anti-Virus doing it’s job currently and catching the file. This did reveal some minor visual glitches with this software on Ubuntu. However thankfully nothing too major.



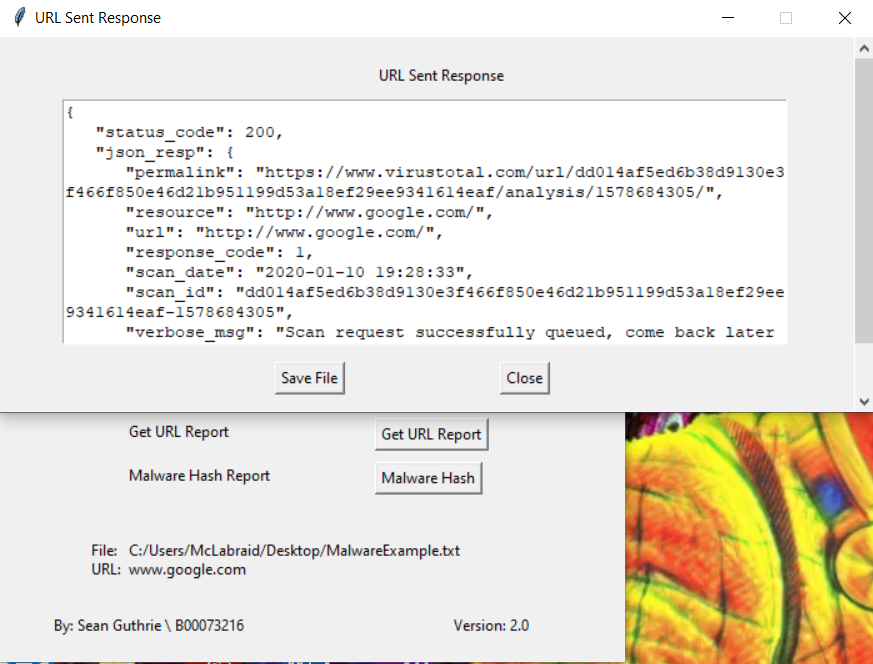
From here we can see that the application was successful in catching the “Virus”.

### Upload URL

From here we can go to the “Upload URL” option on the menu. From here we are greeted by a small window asking for URL, with a small text box and an enter button.



Upon entering the URL, we press the enter button and a window containing the response is sent to the user (It should also be noted that the user can press Enter to submit the URL instead of pressing Enter).

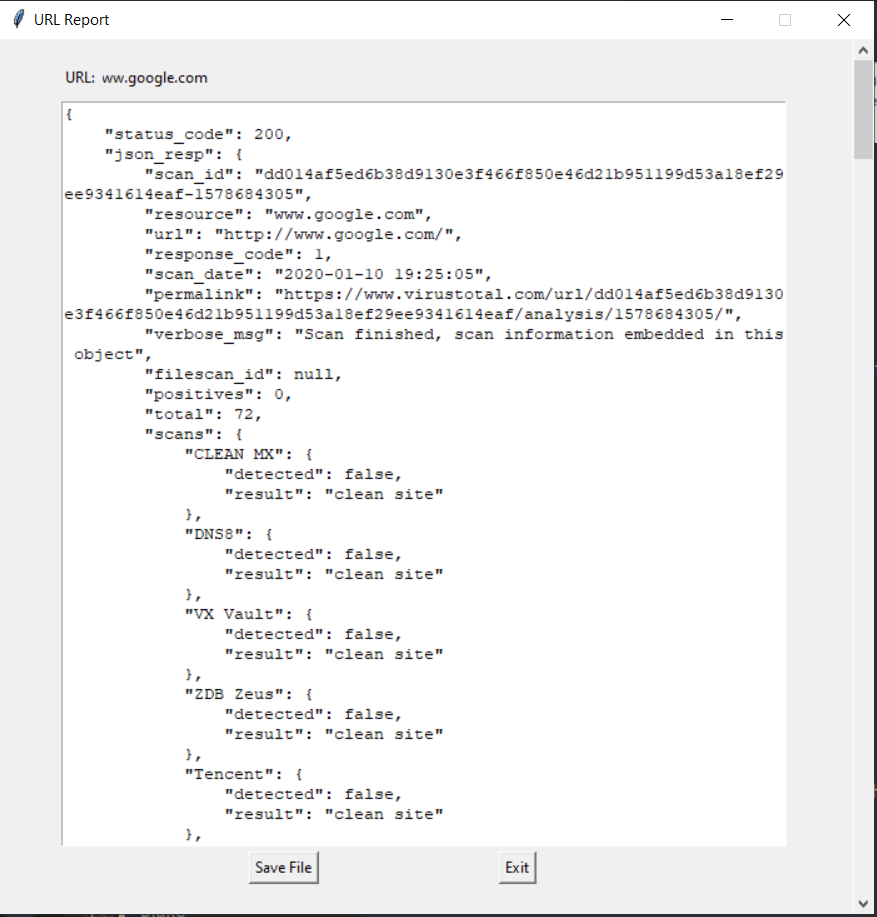


Just like the File response, we also have the option here to save the response(which will also open a save window) or just close the window, and just like when we uploaded the file, the URL is also present on the main menu as a reminder.



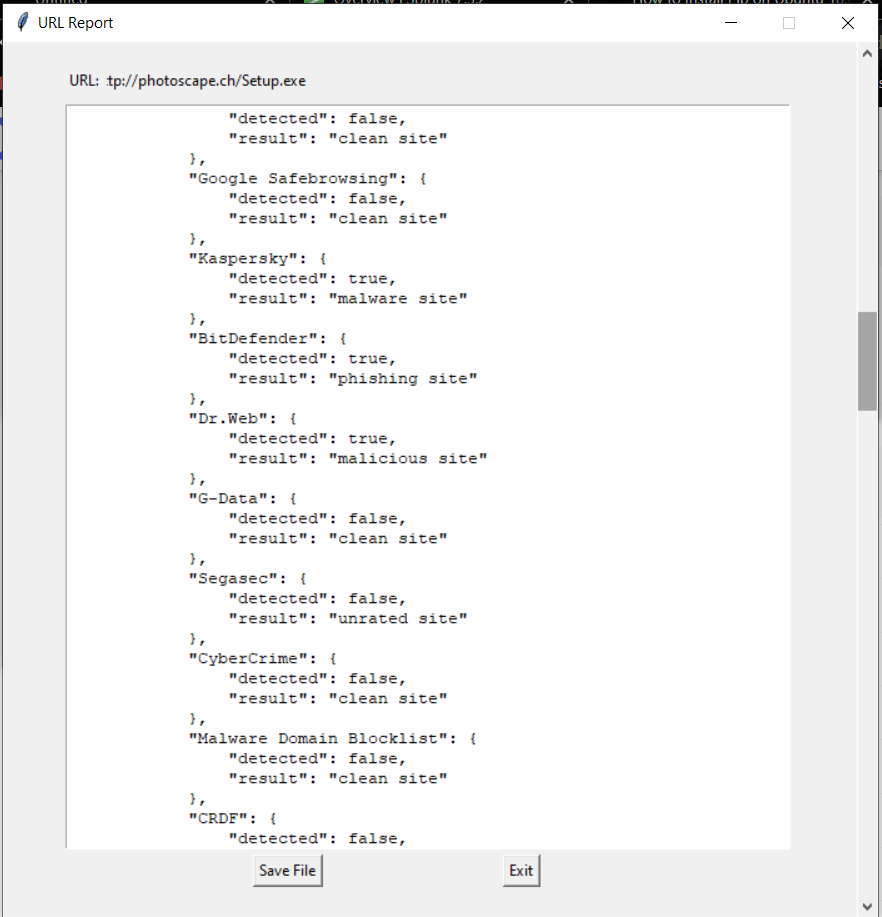
### Get URL Report

The next options on the main menu is the “Get URL Report”, this option produces a window, containing the report for the submitted URL. This file can also be saved and/or close.



### Malicious URL Example

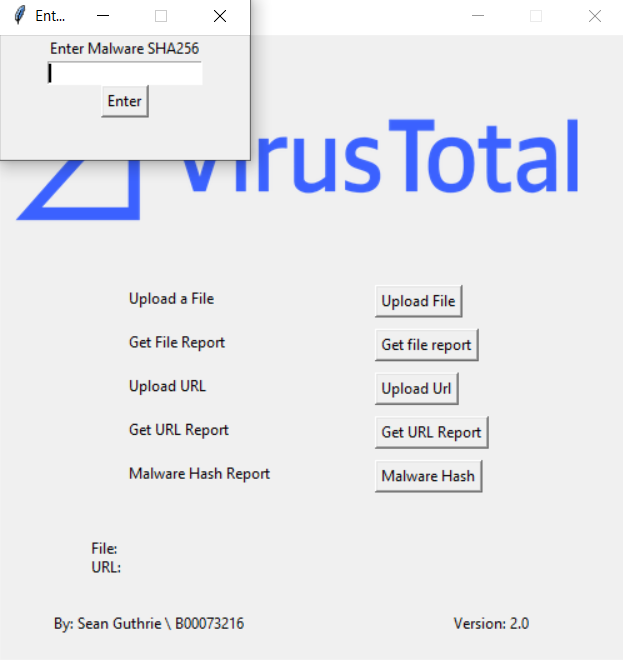
For this section, we used a malicious URL found on a website that collects malicious URL, this URL being: photoscape[.]ch/Setup[.]exe, This URL is known to contain a Trojan virus.



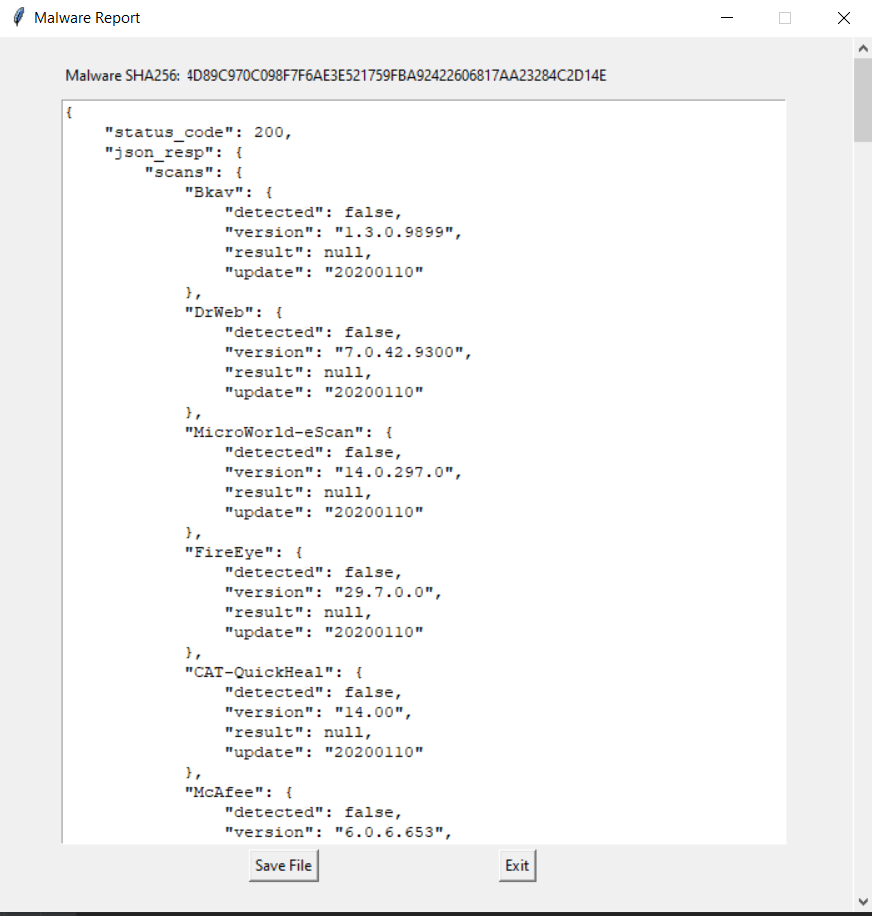
We can see here that the application was successful spotting the malicious site.

### Malware Hash

Lastly, we have the “Malware Hash” option, which allows users to enter a SHA256 into a window.

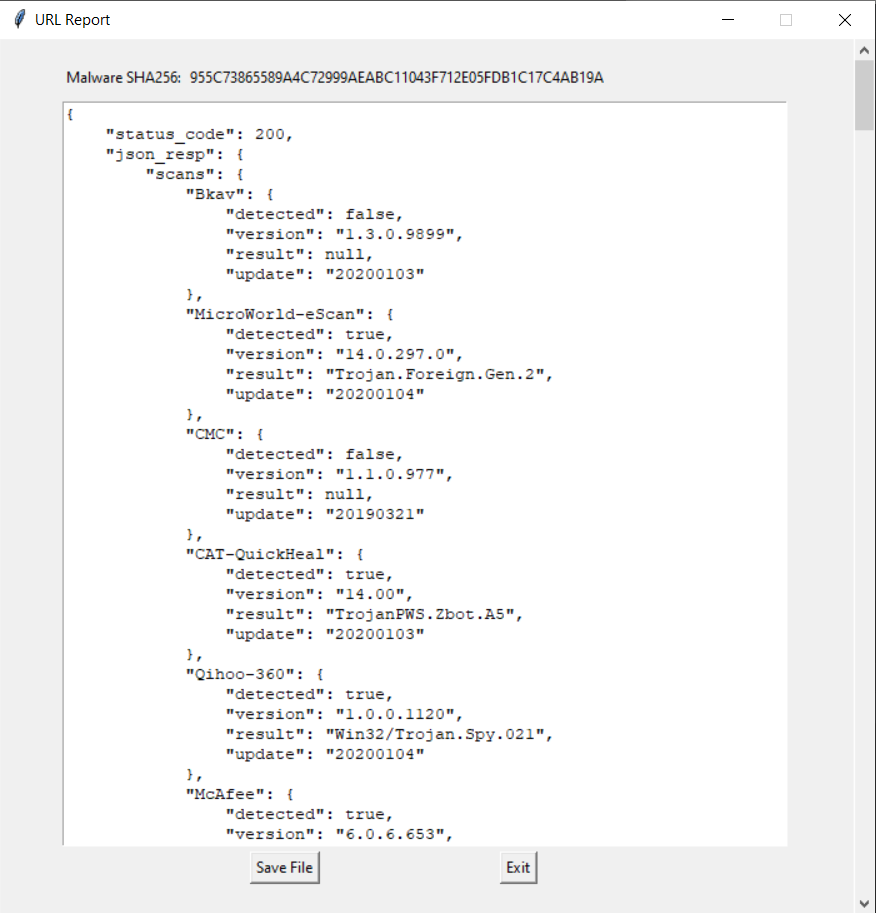


When you submit a Malware SHA256, a window is opened containing a report of the malware. Just like the other reports, we are able to save the report and/or close this window



### Malicious Malware Hash Example

To prove that our Malware Hash program works, we have provided a known hash of a trojan virus, known as the "Trojan.Zbot.Win32.21", found on a site with most known viruses. When we put the hash in (8B2E701E91101955C73865589A4C72999AEABC11043F712E05FDB1C17C4AB19A) the report told us that this Trojan had been detected.

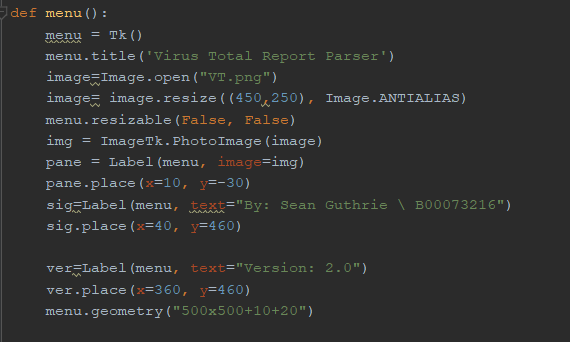


# Code Walkthrough

For this section, we will be giving a walkthrough of how the code working inside of the program, as to give a broader understanding of how this program works. As previously mentioned, this entire application was built in Python 3.6. This program was also split into different function, with each function providing the functionality for each button within the application.

### Main Menu Function

The main menu was contained within the main() function, within the menu() function there is the code for the main GUI and the button menu. This GUI was built with the use of Tkinter library. There is also the code for the VirusTotal Image at the top of the application. This image was handled with this PIL library.



For the menu, we used several button calls, that when clicked would activate the responding function. An example of this would be the button for Uploading the file. The button has a command within its brackets for: “command:lamda: ScanF(FStr)”

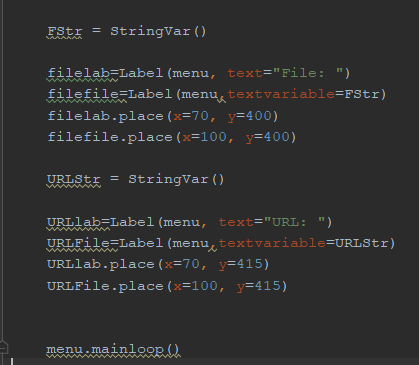
The lamda is a command to initate this command immediate, while the ScanF is calling the “Upload File” function within the program, and the “FStr” is an escape string, used to add the file location to the Label.

This section also contains the positioning code for Tkinter. As to align the buttons in the correct place.



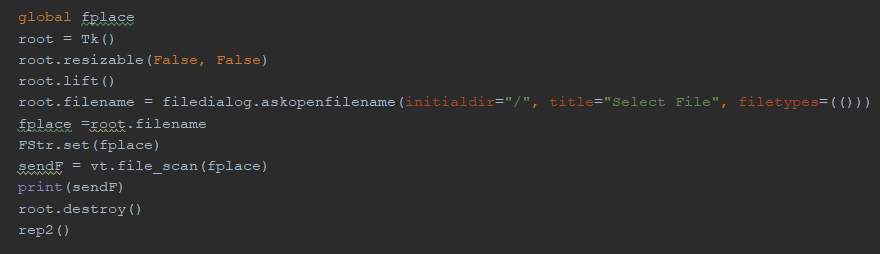
Lastly in this function, we have the two labels that appear on the end of the program, these two being the location of the File and the URL entered into the program. This is done by calling the string that exits the respective functions and being called by this section.

We also have the menu.mainloop() which puts all the Tkinter commands together.

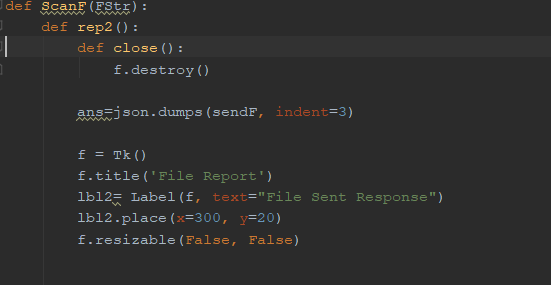


### Upload File Function

The upload File section is contained within the ScanF Function within the code. This section works by creating a Tkinter GUI, that calls on a askopenfinename, which is used to call the window to upload the file, which is saved as root.filename. We then change it to fplace because root.fileplace is too long to remember. We then send it to the vt.file\_scan, which is part of the virustotal\_python library. We then close the window and call the rep2() function which is within the scanF() function.

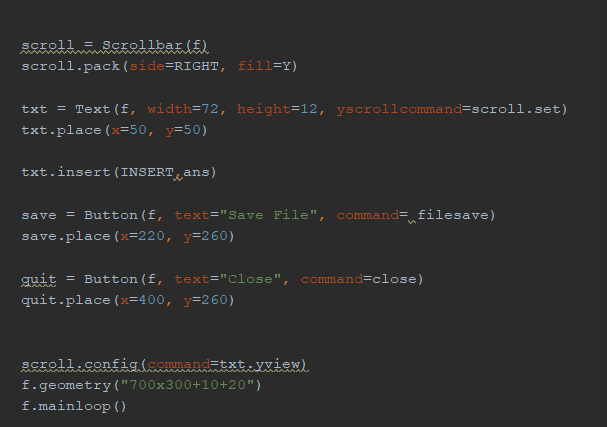


This section creates a GUI report, known as a “File Report” which prints out the response of the upload. This is done by calling the json.dumps command, which dumps the report into a JSON file which can be called



After this section we also have a scrollbar, which allows the user to scroll down a document if it is bigger than the text box, this is done with the Scrollbar(f) command. The Text command also enables the white textbox that is scrollable, with the .insert putting the json.dumps file within the textbox.

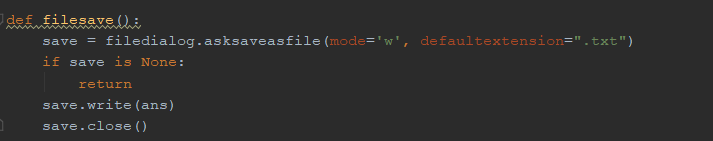
There’s also the save button, that uses the filesave() function, which we’ll look at later. There’s also the quit command which calls the close function that kills the window and brings us back to the menu.



### Save File Function

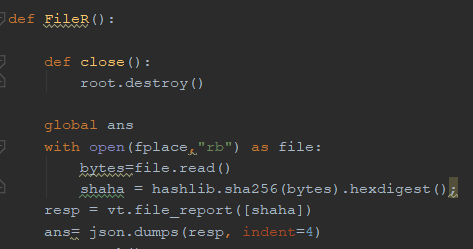
The save file function is a simple command that simply calls for a window to save the file, with the mode of w, as to write the file and the defaultextension of .txt. as to save the file as a text file.

The save.write is then calls the file to be saved, and then closes itself with .close()

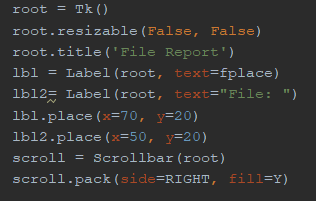


### Read File Function

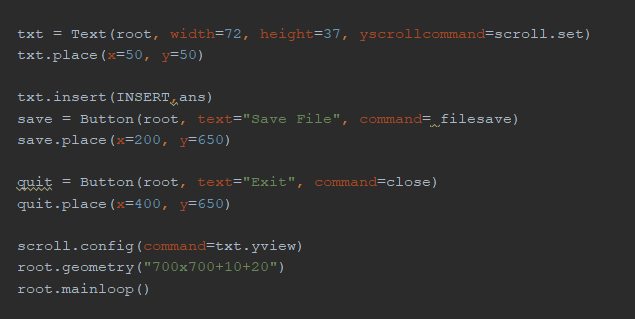
This function is called as FileR(), and is used to get the report for the previously uploaded file. To preform this, the function calls on global fplace variable in the ScanF() function. Getting the Sha256 of this file and sending it to VirusTotal with the vt.file\_report command.



We then build a Tkinter GUI called “File Report”, giving it a scroll bar to allow scrolling.



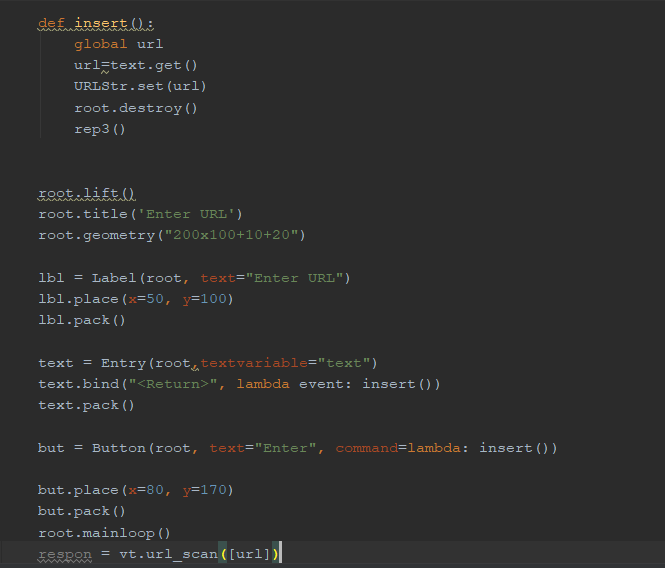
We also have the json.dump appear in text section, giving it the scroll bar for scrolling. Giving it a save button and a close button.



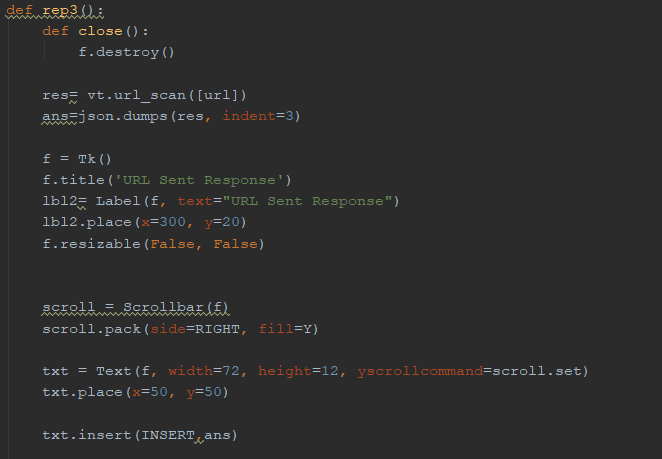
### Scan URL Function

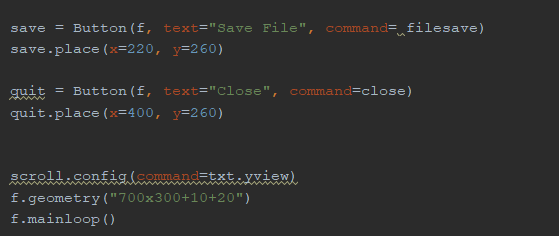
For scanning URLs, we use the ScanURL() function, this section is for getting a URL and scanning it with the VirusTotal API with the vt.url\_scan command. This section creates a GUI that takes in text. It also has an enter button which calls on the insert() function, which sets the url to the text, sets up the function that will go outside the function, and destroys the window.. At the end of the insert command, the rep3() function is called, which produces a report on the success of the upload.

There is also a command in the ScanURL() function that when the enter key is pressed, it will act as the Enter button.



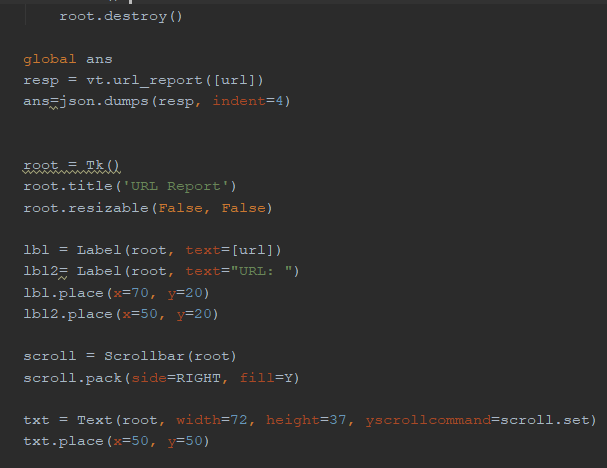
The Rep3() function works very similarly to the previous examples of success reports, it creates a GUI with Tkinter, contains a Scroll bar. Prints the Json file into a text box and allows to save the file or exit the window.

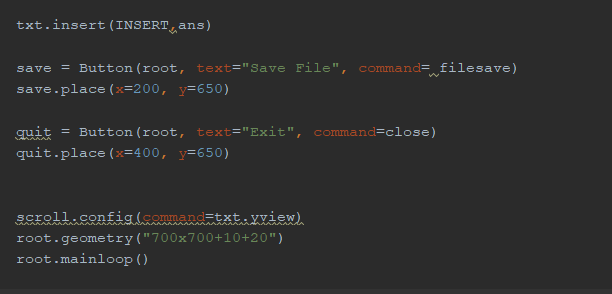




### URL Report Function

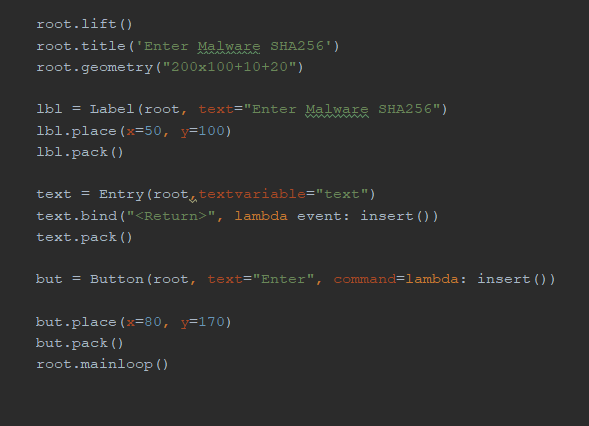
This section produces a report of the submitted URL in the previous section, it does this by calling the URL from the previous variable and getting the report from VirusTotal with the vt.url\_report command. This function starts off by creating the Tkinter GUI, putting the json file into the text box, having a scroll bar, providing a save function, and a close function.



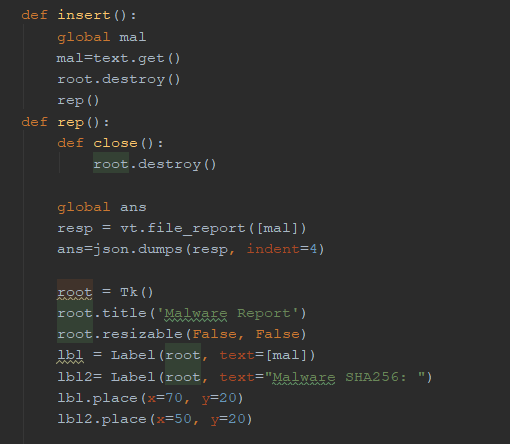


### Malware Hash Function

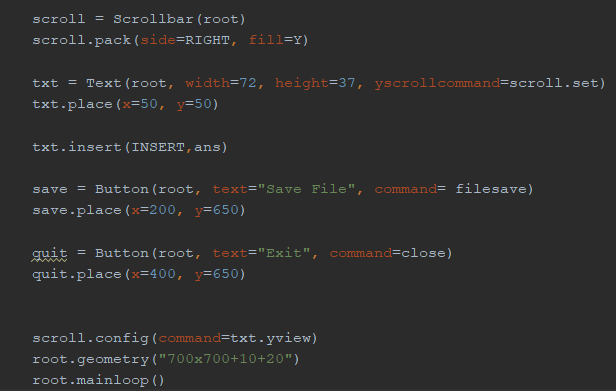
For this section we take the SHA256 hash and send it to Virustotal to scan. We do this in the RepStat(). This section works like the upload URL section, however instead of the vt.url\_upload, we use vt.file\_report to get a report of the malware. This section uses the same type of input box style, with the same enter button feature. That when clicked, leads us to the insert() function.



From here, the text is set to the mal variable, and the mal variable is set to global, the window is then destroyed and the rep() function is called. The rep() function calls the mal variable and sends it to Virustotal, where the report is sent back, it is then converted to a JSON file and put into the text box like previous examples. A GUI is also created with the title “Malware report”



There is also all the same functions as the other reports, a scroll bar for the textbox, a save button and a close button.



# Unresolved bugs

During the course of this assignment, there was two bugs that stuck out. One being an overflow bug and the other being the inability to make a parse file.

The overflow bug seems to trigger when to many variables are passed into the program. The only way to fix this is to reset the application. Due to time constraints, this could not be fixed.

Secondly, for some strange reason, I was unable to parse the data coming from the virustotal API, due to this the data was printed in it’s original format, however it comes out at least structured.

# Conclusions

This tool is useful in getting information about files, URL and Hashes from Virustotal, and for getting reports on all of these respectively. This has defiantly been an interesting project to build and has definitely help me improve my ability with Python. Such as helping me learn how to build a GUI, and to understand API implementation.

# References

VirusTotal(2020) Getting Started. [Online] Available at: https://developers.virustotal.com/reference#getting-started [Accessed 20th Dec 2019]

PyPI. (2020). virustotal-python. [Online] Available at: https://pypi.org/project/virustotal-python/ [Accessed 20th Dec. 2019].

Tutorialspoint(2020). Python – GUI Programming (Tkinter) [Online] Available at: https://www.tutorialspoint.com/python/python\_gui\_programming.htm [Accessed 20th Dec. 2019].

Readthedocs(2020). Pillow(PIL Fork) [Online] Available at: https://pillow.readthedocs.io/en/stable/ [Accessed 30th Dec. 2019].

MetaDefender(2020).Top Infected Hashes[Online] Available at: https://onlinehelp.opswat.com/mdcloud/Top\_infected\_hashes.html [Accessed 6th Jan. 2020].

Malware Domain(2020). Malware Domain List[Online] Available at: https://www.malwaredomainlist.com/mdl.php [Accessed 9th Jan. 2020]

Ikarus.(2020). Download Test Viruses[Online] Available at: https://www.ikarussecurity.com/en/private-customers/download-test-viruses/ [Accessed 10 Jan. 2020]