



References:

- [1] <https://virustotal.github.io/>
- [2] <https://www.virustotal.com/>
- [3] https://en.wikipedia.org/wiki/Antivirus_software
- [4] Krebs, Brian (March 9, 2007). *"Online Anti-Virus Scans: A Free Second Opinion"*. *The Washington Post*. Retrieved February 24, 2011.
- [5] <https://antivirus.comodo.com/faq/how-antivirus-works.php>
- [6] <https://learn.microsoft.com/en-us/connectors/virustotal/>
- [7] <https://developers.virustotal.com/reference/errors>
- [8] <https://developers.virustotal.com/reference/>

What is ANTIVIRUS?

Antivirus software (abbreviated to AV software), also known as anti-malware, is a computer program used to prevent, detect, and remove malware [4].

What is Malware?

Malware is any type of software written with malicious intent, which can range from data theft, computer damage, or general privacy invasion. Malware tends to be spread when a suspicious link or download is opened by a victim. Ranging anywhere from viruses to ransomware, malware is an umbrella term that encompasses many types.

Online Scanning

Some antivirus vendors maintain websites with free online scanning capability of the entire computer, critical areas only, local disks, folders or files. Periodic online scanning is a good idea for those that run antivirus applications on their computers because those applications are frequently slow to catch threats. One of the first things that malicious software does in an attack is disable any existing antivirus software and sometimes the only way to know of an attack is by turning to an online resource that is not installed on the infected computer [4].

How it works?

Antivirus software scans a file, program, or an application and compares a specific set of code with information stored in its database. If it finds code that is identical or similar to a piece of known malware in the database, that code is considered malware and is quarantined or removed [5].

Initial Steps

Getting to know the websites presenting free scan on files

- VirusTotal

Virus Total is an online service that analyzes suspicious files and URLs to detect types of malwares and malicious content using antivirus engines and website scanners. It provides an API that allows users to access the information generated by VirusTotal [6]. Some difference between the Community and Premium version of this software is as follow.

! Public API constraints and restrictions

The Public API is limited to 500 requests per day and a rate of 4 requests per minute.

The Public API must not be used in commercial products or services.

The Public API must not be used in business workflows that do not contribute new files.

You are not allowed to register multiple accounts to overcome the aforementioned limitations.

Premium API highlights

The Premium API does not have request rate or daily allowance limitations, limits are governed by your licensed service step.

The Premium API returns more threat context and exposes advanced threat hunting and malware discovery endpoints and functionality.

The Premium API is governed by an SLA that guarantees readiness of data.

Detail description of difference can be found here at:

<https://developers.virustotal.com/reference/public-vs-premium-api>

Steps to use its API:

1. Sign up/in at <https://www.virustotal.com/gui/join-us>
2. All we need is the personal API key which can be found in the personal setting section.

API Key

Request premium API key

View API key

API Key: *****

This is your personal key. Do not disclose it to anyone that you do not trust, do not embed it in scripts or software from which it can be easily retrieved if you care about its confidentiality. By submitting data using your API key, you are agreeing to our [Terms of Service](#) and [Privacy Policy](#), and to the [sharing of your Sample submissions with the security community](#). Please do not submit any personal information; VirusTotal is not responsible for the contents of your submissions. [Learn more](#)

3. In order to start using the API, you can easily choose your programming language or the way you want to use the API and get some helps. As we are working with Python programming language, we chose Python client and then directed to <https://virustotal.github.io/vt->

4 Network Security Final Project – Antivirus

py/howtoinstall.html where we learnt how to install all the dependencies and requirement. Moreover, we read and used their sample examples taught at <https://github.com/VirusTotal/vt-py/tree/master/examples> as help.

Some Example of VirusTotal

- Getting information about file

```
Examples > GetInformationAboutFile.py > ...
1  import vt
2
3  def main():
4      # Enter your own API key as parameter
5      client = vt.client.Client("51bd951cd29384782a40f883531b182a06adb725331ea8c38b7b1f00e45826ca")
6
7      # Ask for the file you are interested in, you can replace the hash in the example with some other SHA-256, SHA-1 or MD5
8      file = client.get_object("/files/44d88612fea8a8f36de82e1278abb02f")
9
10     print(file.size)
11     print(file.sha256)
12     print(file.type_tag)
13
14
15 if __name__ == '__main__':
16     main()
```

- Getting information about URL

```
Examples > GetInformationAboutURL > ...
1  import vt
2
3  def main():
4      client = vt.client.Client("51bd951cd29384782a40f883531b182a06adb725331ea8c38b7b1f00e45826ca")
5
6      url_id = vt.url_id("https://www.google.com")
7      url = client.get_object('/urls/{', url_id)
8
9      print(url.times_submitted)
10     print(url.last_analysis_stats)
11
12 if __name__ == "__main__":
13     main()
```

5 Network Security Final Project – Antivirus

○ Scanning File

```
Examples > ScanURL.py > ...
1  import vt
2  import time
3
4  def main():
5      # Enter your own API key as parameter
6      client = vt.client.Client("51bd951cd29384782a40f883531b182a06adb725331ea8c38b7b1f00e45826ca")
7
8      analysis = client.scan_url('https://www.google.com/')
9
10     while True:
11         analysis = client.get_object("/analyses/{}", analysis.id)
12         print(analysis.status)
13         if analysis.status == "completed":
14             break
15         time.sleep(30)
16
17
18 if __name__ == '__main__':
19     main()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Python + - [] [] ^ x

```
PS C:\Users\ALSHO\Downloads\Network Security\Project\src> & C:/Python310/python.exe "c:/Users/ALSHO/Downloads/Network Security/Project/src/Examples/ScanURL.py"
completed
PS C:\Users\ALSHO\Downloads\Network Security\Project\src>
```

○ Scanning an URL

```
Examples > ScanFile.py > ...
1  import vt
2  import time
3
4  def main():
5      # Enter your own API key as parameter
6      client = vt.client.Client("51bd951cd29384782a40f883531b182a06adb725331ea8c38b7b1f00e45826ca")
7
8
9      with open("DocToScan\IoTArticle2.pdf", "rb") as file:
10         analysis = client.scan_file(file)
11
12     while True:
13         analysis = client.get_object("/analyses/{}", analysis.id)
14         print(analysis.status)
15         if analysis.status == "completed":
16             break
17         time.sleep(30)
18
19
20 if __name__ == '__main__':
21     main()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Python + - [] [] ^ x

```
PS C:\Users\ALSHO\Downloads\Network Security\Project\src> & C:/Python310/python.exe "c:/Users/ALSHO/Downloads/Network Security/Project/src/Examples/ScanFile.py"
queued
queued
queued
queued
queued
completed
PS C:\Users\ALSHO\Downloads\Network Security\Project\src>
```

Errors

What we are going to do in this project at its first steps is to combine these examples to create one whole application that can do these tasks for us. One thing that we should consider is about errors. Sometimes, users may do not enter a valid IP address. In these times, when the API sends our data to the server in order to scan them, it receives errors and this errors return to the application. In order to create a well application we must firstly familiarize with these errors and then try to tackle them through some steps or even asking user to check them or even ignore these errors to prevent the program from crashing. The below table shows the possible errors that we may face during execution. The table is based on what is defined in the [7].

HTTP Code	Error code	Description
400	BadRequestError	The API request is invalid or malformed. The message usually provides details about why the request is not valid.
400	InvalidArgumentError	Some of the provided arguments are incorrect.
400	NotAvailableYet	The resource is not available yet, but will become available later.
400	UnselectiveContentQueryError	Content search query is not selective enough.
400	UnsupportedContentQueryError	Unsupported content search query.
401	AuthenticationRequiredError	The operation requires an authenticated user. Verify that you have provided your API key.
401	UserNotActiveError	The user account is not active. Make sure you properly activated your account by following the link sent to your email.
401	WrongCredentialsError	The provided API key is incorrect.
403	ForbiddenError	You are not allowed to perform the requested operation.
404	NotFoundError	The requested resource was not found.
409	AlreadyExistsError	The resource already exists.
424	FailedDependencyError	The request depended on another request and that request failed.
429	QuotaExceededError	You have exceeded one of your quotas (minute, daily or monthly). Daily quotas are reset every day at 00:00 UTC. You may have run out of disk space and/or number of files on your VirusTotal Monitor account.
429	TooManyRequestsError	Too many requests.
503	TransientError	Transient server error. Retry might work.
504	DeadlineExceededError	The operation took too long to complete.

Implementation

Since the project consists of various areas and it needs various methods, we'll do our best to implement it with in various files to help the readability and become more organized.

❖ URLs Module

This module consists of methods that can be used to scan the given URL. It simply gets API key and the URL that we want to scan and returns a JSON format data describing the situation of that given website.

Scan Method

By calling this method, it uses the given API and URL that it took as input when you were creating an object of its class and returns the results of scanning that URL. The code is as follow:

```
def scan(self):
    ParsedURL = urllib.parse.quote(self.ScanURL, safe='') # by default safe skip '/' so we needed to emptied it
    payload = f"url={ParsedURL}"
    headers = {
        "accept": "application/json",
        "x-apikey": f"{self.api}",
        "content-type": "application/x-www-form-urlencoded"
    }
    self.ScanResponse = requests.post(self.MainURL, data=payload, headers=headers)
    return self.ScanResponse # In order to check the response you should use .text
```

The returned response is an object and if you keen to check it out, you must call its instance by object.text in order to see the result data which is in our case looks like the below picture:

```
Scanning...
{
  "data": {
    "type": "analysis",
    "id": "u-d0e196a0c25d35dd0a84593cbae0f38333aa58529936444ea26453eab28dfc86-1673025429"
  }
}
Scanning finished!
```

The result data usually describes the job we did and tell us the “id” of that job in order to check the detailed report of that job.

Rescan Method

This method does like the previous one with only one difference on the thing it checks. This method rescans the given URL, it doesn't take URL again, but it does create new ID since it does new scan. Basically, it uses the ID of a scan that formerly has been done and rescan that data or URL.

```
def rescan(self):
    self.RescanURL = f"{self.MainURL}/{self.id}/analyse"
    headers = {
        "accept": "application/json",
        "x-apikey": f"{self.api}"
    }
    self.ReScanResponse = requests.post(self.RescanURL, headers=headers)
    return self.ReScanResponse # In order to check the response you should use .text
```

The response data of this method is as follow:

```
Rescanning!
{
  "data": {
    "type": "analysis",
    "id": "u-e5080dce22824e5ce2239feb07c87b9009f974d8793a5e227a7a6393d2e05883-1673025431"
  }
}
Rescan finished!
```

You can see that the details of this scan are now have another ID which can be use to get more information and report for this scan.

Report Method

This method returns the report of the previous scans and gives information about the results of the scans. Since we had two scan (first scan and rescan), we considered a bool parameter for this method which helps the user to select which previous scan he wants to check.

```
def report(self, FirstScan=True):
    if FirstScan:
        # First we get the ID from the scan's response output
        self.id = json.loads(self.ScanResponse.text)['data']['id']
    else:
        self.id = json.loads(self.ReScanResponse.text)['data']['id']
    self.id = base64.urlsafe_b64encode(self.id.encode()).decode().strip("=")
    reportURL = f"{self.MainURL}/{self.id}"
    print(reportURL)
    headers = {
        "accept": "application/json",
        "x-apikey": f"{self.api}"
    }
    self.ReportResponse = requests.get(reportURL, headers=headers)
    return self.ReportResponse # In order to check the response you should use .text
```

Part of its output is as follow:

```
Reporting!
https://www.virustotal.com/api/v3/urls/c2VsZi5pZA
{
  "data": {
    "attributes": {
      "last_modification_date": 1673026562,
      "times_submitted": 34,
      "total_votes": {
        "harmless": 0,
        "malicious": 0
      },
    },
    "threat_names": [],
    "redirection_chain": [
      "http://self.id/"
    ],
    "last_submission_date": 1673026551,
    "last_http_response_content_length": 29698,
    "last_http_response_headers": {
      "Permissions-Policy": "interest-cohort=()",
      "X-Powered-By": "Next.js",
      "Transfer-Encoding": "chunked",
      "Age": "0",
      "Strict-Transport-Security": "max-age=63072000",
      "Server": "Vercel",
      "Cache-Control": "private, no-cache, no-store, max-age=0, must-revalidate",
      "Connection": "keep-alive",
      "X-Vercel-Cache": "MISS",
      "X-Matched-Path": "/",
      "Date": "Fri, 06 Jan 2023 17:35:52 GMT",
      "Content-Type": "text/html; charset=utf-8",
      "Content-Encoding": "gzip",
      "X-Vercel-Id": "cle1::iad1::ms7hg-1673026552732-d26d3b40f619"
    },
  },
}
```

❖ Files Module

Like what we did previously, here, again, we designed a module consisting of methods scan and report which this time concern to files. Since the community version of VirusTotal has a limitation on the amount of calling for rescan, we avoid implementing rescan method here. However, you can find the instruction of here at <https://developers.virustotal.com/reference/files-analyse> . Moreover, before talking about the methods, it worth to mention that if we want to work with files which sizes are less than 32MB, we can simply use Post method and send our file for scanning, but if there were a need to scan a file which size is bigger than 32MB, you can follow the instructions described here at <https://developers.virustotal.com/reference/files-upload-url> .

Scan Method

This method works like the URL scan method. It uses Post method to send the file which has been read from the input in binary mode. The next picture demonstrates its codes:

```
def scanUpload(self):
    files = {"file": ("IoTArticle1.pdf", open(self.ScanFile, "rb"), "application/pdf")}
    headers = {
        "accept": "application/json",
        "x-apikey": self.api,
    }
    self.ScanResponse = requests.post(self.MainURL, files=files, headers=headers)
    return self.ScanResponse
```

Having said that, scan method also works as getting file from input and in the response it tells us about the ID of the scanned file.

```
Scanning File....
{
  "data": {
    "type": "analysis",
    "id": "NGY0NzMzODNjYTUxMjE0NTljZDQ2ZWNmNDJjOWZjZGU6MTY3MzA0MTk4Mg=="
  }
}
Scanning File finished!
```

Report Method

Like before, this method reports the status of the given file. There is **BIG** notation about this method. In order to access the detail descriptions, here, we don't use the ID of the file, but we use a hashing method. In the background, when the VirusTotal receives the file, it generates a hash code. To access the descriptions of the files we need to first use a hashing method in order

11 Network Security Final Project – Antivirus

to get the hash values of the file and then use that hash code to access the file in the servers of VirusTotal.

Here is the Hash method. This method generates SHA-256 from the input file.

```
def hashTheFile(self):
    sha256_hash = hashlib.sha256()
    with open(self.ScanFile, "rb") as f:
        # Read and update hash string value in blocks of 4K
        for byte_block in iter(lambda: f.read(4096), b''):
            sha256_hash.update(byte_block)
    return sha256_hash.hexdigest()
```

Here is the report method.

```
def report(self):
    HashID = self.hashTheFile()
    reportURL = f"{self.MainURL}/{HashID}"
    print("Link: ", reportURL)
    headers = {
        "accept": "application/json",
        "x-apikey": f"{self.api}"
    }
    self.ReportResponse = requests.get(reportURL, headers=headers)
    return self.ReportResponse # In order to check the response you should use .text
```

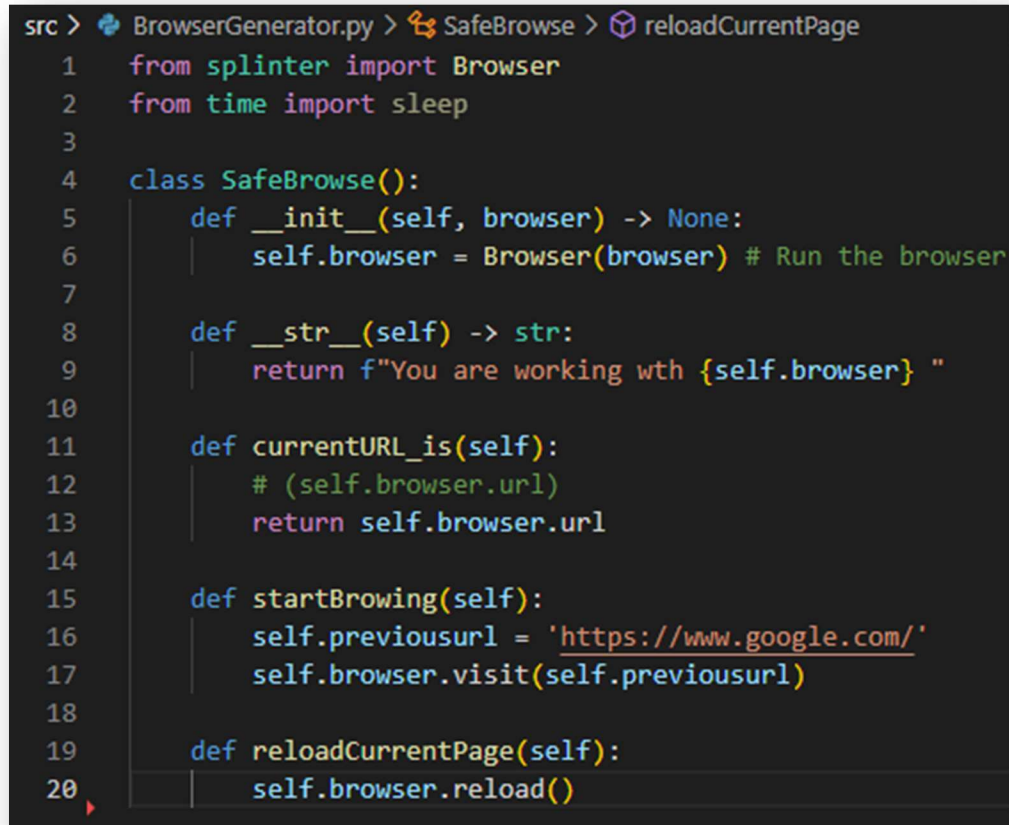
12 Network Security Final Project – Antivirus

The below picture is a part of the output that report method returns.

```
Reporting!
Link: https://www.virustotal.com/api/v3/files/d2a31eee1d5f79edc68b98700c71c875b329d4dcccc3ee3d08995dcc7c0960f3
{
  "data": {
    "attributes": {
      "type_description": "PDF",
      "tlsh": "T179F4AE1025E176BD9EFE9F3C1182EA1426CB31B2B8CB15A3F99F1C58FF50915D0AE285",
      "vhash": "9cfb48dfdd78d3703f08e0d4a1d9971cb",
      "trid": [
        {
          "file_type": "Adobe Portable Document Format",
          "probability": 100.0
        }
      ],
      "crowdsourced_yara_results": [
        {
          "description": "This signature detects an Adobe Type 1 Font. The Type 1 Font Format is a standardized font format for digital imaging applications.",
          "source": "https://github.com/InQuest/yara-rules-vt",
          "author": "InQuest Labs",
          "ruleset_name": "Adobe_Type_1_Font",
          "rule_name": "Adobe_Type_1_Font",
          "ruleset_id": "0124227417"
        },
        {
          "description": "This signature identifies Adobe Extensible Metadata Platform (XMP) identifiers embedded within files. Defined as a standard for mapping graphical asset relationships, XMP allows for tracking of both parent-child relationships and individual revisions. There are three categories of identifiers: original document, document, and instance. Generally, XMP data is stored in XML format, updated on save/copy, and embedded within the graphical asset. These identifiers can be used to track both malicious and benign graphics within common Microsoft and Adobe document lures.",
          "source": "https://github.com/InQuest/yara-rules-vt",
          "author": "InQuest Labs",
          "ruleset_name": "Adobe_XMP_Identifier",
          "rule_name": "Adobe_XMP_Identifier",
          "ruleset_id": "0121ae37cc"
        }
      ]
    }
  },
  ],
```

❖ BrowserGenerator Module

For safe browsing, I implemented another module, which opens a browser's window. Users can simply enter websites with this browser. This browser takes the URL of the sites that users entered and scan it and shows the results in the output. Whenever user opens another link in the current tab, the code scans the URL and shows the report in the terminal. The following picture shows the code for browser.



```
src > BrowserGenerator.py > SafeBrowse > reloadCurrentPage
1  from splinter import Browser
2  from time import sleep
3
4  class SafeBrowse():
5      def __init__(self, browser) -> None:
6          self.browser = Browser(browser) # Run the browser
7
8      def __str__(self) -> str:
9          return f"You are working with {self.browser} "
10
11     def currentURL_is(self):
12         # (self.browser.url)
13         return self.browser.url
14
15     def startBrowsing(self):
16         self.previousurl = 'https://www.google.com/'
17         self.browser.visit(self.previousurl)
18
19     def reloadCurrentPage(self):
20         self.browser.reload()
```

As it is obviously shown in the picture, whenever user creates an object with this module, the browser starts automatically. Browser parameter in the “init” method is the name of the browser that user wants to open. It is either “chrome”, or “firefox”.

Methods Description

- currentURL_is() : This method returns the current URL watching by user
- startBrowsing(): This method is just for the initial step. It opens google by default
- reloadCurrentPage(): This method reloads the current site