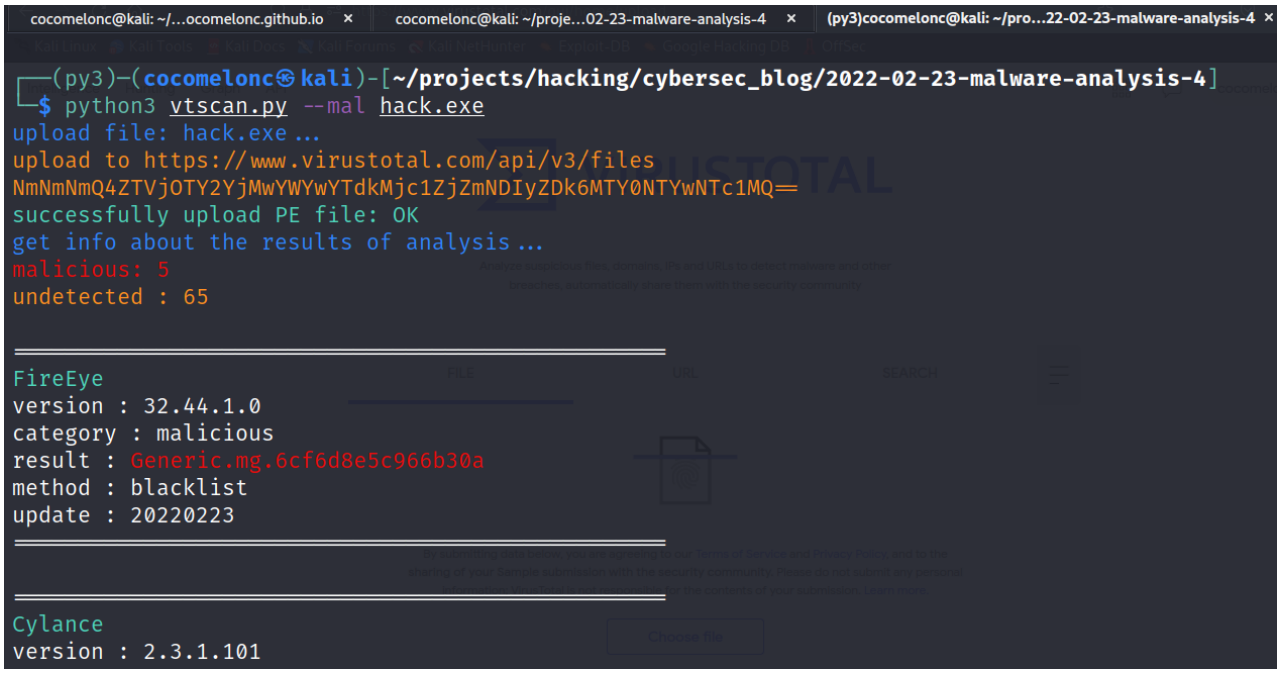
Malware analysis 4: Work with VirusTotal API v3. Create own python script.

cocomelonc.github.io/tutorial/2022/02/23/malware-analysis-4.html

February 23, 2022

7 minute read

Hello, cybersecurity enthusiasts and white hackers!

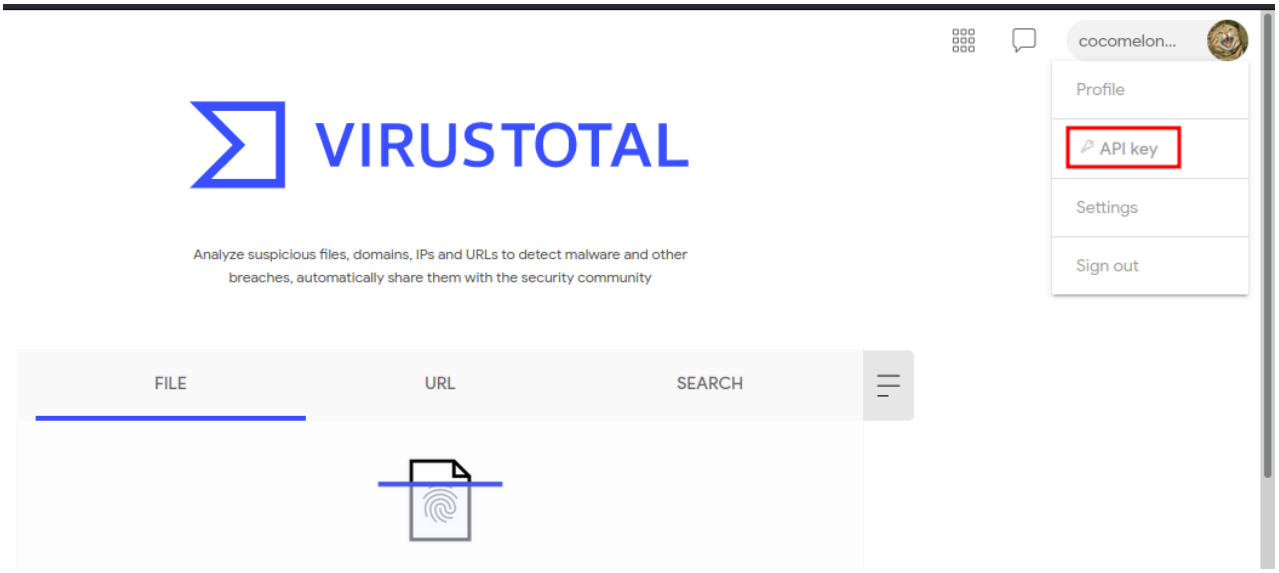


This post is the result of my own research on how the VirusTotal API works.

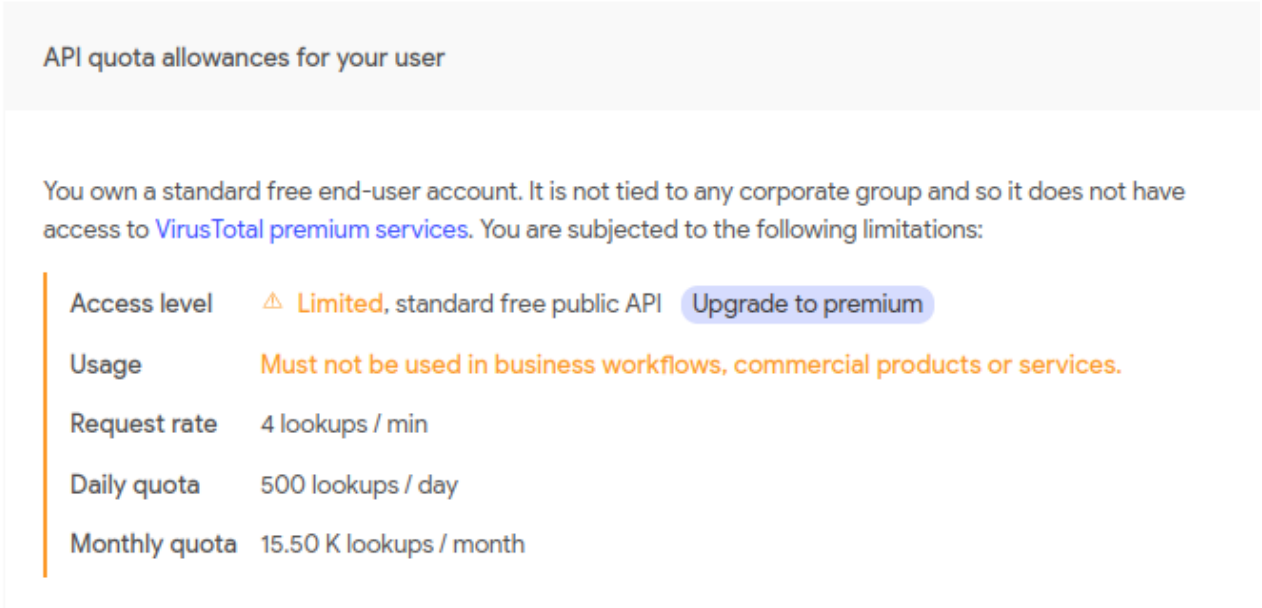
about VirusTotal API

You have probably used the services of the https://virustotal.com site more than once to check whether the binaries contain malicious functions, or to test your own developments.

This service have a free API. After registering an account, you can get an API key:



But there is a limitation that is not critical for research purposes: 4 requests per minute:



API documentation is located at here

practical example

For practical purposes, I wrote a simple script that uploads a local file to VirusTotal (so far only less than `32 MB`) and gives the result of its analysis in numbers, how many AV-engines consider it malicious.

Let's go. Firstly, according to the documentation, we indicate our API key in all requests:

```
#...

# for terminal colors
class Colors:
    BLUE = '\033[94m'
    GREEN = '\033[92m'
    YELLOW = '\033[93m'
    RED = '\033[91m'
    PURPLE = '\033[95m'
    ENDC = '\033[0m'

VT_API_KEY = "My VirusTotal API key"

#...
class VTScan:
    def __init__(self):
        self.headers = {
            "x-apikey" : VT_API_KEY,
            "User-Agent" : "vtscan v.1.0",
            "Accept-Encoding" : "gzip, deflate",
        }
#...
```

Then I created function which upload local file to virustotal:

```
class VTScan:
#...

    def upload(self, malware_path):
        print (Colors.BLUE + "upload file: " + malware_path + "..." + Colors.ENDC)
        self.malware_path = malware_path
        upload_url = VT_API_URL + "files"
        files = {"file" : (
            os.path.basename(malware_path),
            open(os.path.abspath(malware_path), "rb"))
        }
        print (Colors.YELLOW + "upload to " + upload_url + Colors.ENDC)
        res = requests.post(upload_url, headers = self.headers, files = files)
        if res.status_code == 200:
            result = res.json()
            self.file_id = result.get("data").get("id")
            print (Colors.YELLOW + self.file_id + Colors.ENDC)
            print (Colors.GREEN + "successfully upload PE file: OK" + Colors.ENDC)
        else:
            print (Colors.RED + "failed to upload PE file :(" + Colors.ENDC)
            print (Colors.RED + "status code: " + str(res.status_code) + Colors.ENDC)
            sys.exit()

#...
```

Then, created function which get information about the results of analysis:

```
    def analyse(self):
        print (Colors.BLUE + "get info about the results of analysis..." +
Colors.ENDC)
        analysis_url = VT_API_URL + "analyses/" + self.file_id
        res = requests.get(analysis_url, headers = self.headers)
        if res.status_code == 200:
            result = res.json()
            status = result.get("data").get("attributes").get("status")
            if status == "completed":
                stats = result.get("data").get("attributes").get("stats")
                results = result.get("data").get("attributes").get("results")
                print (Colors.RED + "malicious: " + str(stats.get("malicious")) +
Colors.ENDC)
                print (Colors.YELLOW + "undetected : " + str(stats.get("undetected"))
+ Colors.ENDC)
                print ()
                for k in results:
                    if results[k].get("category") == "malicious":
                        print ("================================================")
                        print (Colors.GREEN + results[k].get("engine_name") +
Colors.ENDC)
                        print ("version : " + results[k].get("engine_version"))
                        print ("category : " + results[k].get("category"))
                        print ("result : " + Colors.RED + results[k].get("result") +
Colors.ENDC)
                        print ("method : " + results[k].get("method"))
                        print ("update : " + results[k].get("engine_update"))
                        print ("================================================")
                        print ()
                print (Colors.GREEN + "successfully analyse: OK" + Colors.ENDC)
                sys.exit()
            elif status == "queued":
                print (Colors.BLUE + "status QUEUED..." + Colors.ENDC)
                with open(os.path.abspath(self.malware_path), "rb") as malware_path:
                    b = f.read()
                    hashsum = hashlib.sha256(b).hexdigest()
                    self.info(hashsum)
        else:
            print (Colors.RED + "failed to get results of analysis :(" + Colors.ENDC)
            print (Colors.RED + "status code: " + str(res.status_code) + Colors.ENDC)
            sys.exit()
```

Here, if the file that we uploaded is analyzed and ready, then we output the result to the console: how many engines consider our file to be malicious in total, if the file is in the queue, then we read the analysis results using the SHA-256 sum of our file as an identifier:

```
    def info(self, file_hash):
        print (Colors.BLUE + "get file info by ID: " + file_hash + Colors.ENDC)
        info_url = VT_API_URL + "files/" + file_hash
        res = requests.get(info_url, headers = self.headers)
        if res.status_code == 200:
            result = res.json()
            if result.get("data").get("attributes").get("last_analysis_results"):
                stats =
result.get("data").get("attributes").get("last_analysis_stats")
                results =
result.get("data").get("attributes").get("last_analysis_results")
                print (Colors.RED + "malicious: " + str(stats.get("malicious")) +
Colors.ENDC)
                print (Colors.YELLOW + "undetected : " + str(stats.get("undetected"))
+ Colors.ENDC)
                print ()
                for k in results:
                    if results[k].get("category") == "malicious":
                        print ("================================================")
                        print (Colors.GREEN + results[k].get("engine_name") +
Colors.ENDC)
                        print ("version : " + results[k].get("engine_version"))
                        print ("category : " + results[k].get("category"))
                        print ("result : " + Colors.RED + results[k].get("result") +
Colors.ENDC)
                        print ("method : " + results[k].get("method"))
                        print ("update : " + results[k].get("engine_update"))
                        print ("================================================")
                        print ()
                print (Colors.GREEN + "successfully analyse: OK" + Colors.ENDC)
                sys.exit()
            else:
                print (Colors.BLUE + "failed to analyse :(..." + Colors.ENDC)

        else:
            print (Colors.RED + "failed to get information :(" + Colors.ENDC)
            print (Colors.RED + "status code: " + str(res.status_code) + Colors.ENDC)
            sys.exit()
```

So, full source code of our tool is:

```
# upload PE file to VirusTotal
# then get info about the results
# of analysis, print if malicious
import os
import sys
import time
import json
import requests
import argparse
import hashlib

# for terminal colors
class Colors:
    BLUE = '\033[94m'
    GREEN = '\033[92m'
    YELLOW = '\033[93m'
    RED = '\033[91m'
    PURPLE = '\033[95m'
    ENDC = '\033[0m'

# VirusTotal API key
VT_API_KEY = "My VirusTotal API key"

# VirusTotal API v3 URL
VT_API_URL = "https://www.virustotal.com/api/v3/"

# upload malicious file to VirusTotal and analyse
class VTScan:
    def __init__(self):
        self.headers = {
            "x-apikey" : VT_API_KEY,
            "User-Agent" : "vtscan v.1.0",
            "Accept-Encoding" : "gzip, deflate",
        }

    def upload(self, malware_path):
        print (Colors.BLUE + "upload file: " + malware_path + "..." + Colors.ENDC)
        self.malware_path = malware_path
        upload_url = VT_API_URL + "files"
        files = {"file" : (
            os.path.basename(malware_path),
            open(os.path.abspath(malware_path), "rb"))
        }
        print (Colors.YELLOW + "upload to " + upload_url + Colors.ENDC)
        res = requests.post(upload_url, headers = self.headers, files = files)
        if res.status_code == 200:
            result = res.json()
            self.file_id = result.get("data").get("id")
            print (Colors.YELLOW + self.file_id + Colors.ENDC)
            print (Colors.GREEN + "successfully upload PE file: OK" + Colors.ENDC)
        else:
            print (Colors.RED + "failed to upload PE file :(" + Colors.ENDC)
```

```
            print (Colors.RED + "status code: " + str(res.status_code) + Colors.ENDC)
            sys.exit()

    def analyse(self):
        print (Colors.BLUE + "get info about the results of analysis..." +
Colors.ENDC)
        analysis_url = VT_API_URL + "analyses/" + self.file_id
        res = requests.get(analysis_url, headers = self.headers)
        if res.status_code == 200:
            result = res.json()
            status = result.get("data").get("attributes").get("status")
            if status == "completed":
                stats = result.get("data").get("attributes").get("stats")
                results = result.get("data").get("attributes").get("results")
                print (Colors.RED + "malicious: " + str(stats.get("malicious")) +
Colors.ENDC)
                print (Colors.YELLOW + "undetected : " + str(stats.get("undetected"))
+ Colors.ENDC)
                print ()
                for k in results:
                    if results[k].get("category") == "malicious":
                        print ("================================================")
                        print (Colors.GREEN + results[k].get("engine_name") +
Colors.ENDC)
                        print ("version : " + results[k].get("engine_version"))
                        print ("category : " + results[k].get("category"))
                        print ("result : " + Colors.RED + results[k].get("result") +
Colors.ENDC)
                        print ("method : " + results[k].get("method"))
                        print ("update : " + results[k].get("engine_update"))
                        print ("================================================")
                        print ()
                print (Colors.GREEN + "successfully analyse: OK" + Colors.ENDC)
                sys.exit()
            elif status == "queued":
                print (Colors.BLUE + "status QUEUED..." + Colors.ENDC)
                with open(os.path.abspath(self.malware_path), "rb") as malware_path:
                    b = f.read()
                    hashsum = hashlib.sha256(b).hexdigest()
                    self.info(hashsum)
        else:
            print (Colors.RED + "failed to get results of analysis :(" + Colors.ENDC)
            print (Colors.RED + "status code: " + str(res.status_code) + Colors.ENDC)
            sys.exit()

    def run(self, malware_path):
        self.upload(malware_path)
        self.analyse()

    def info(self, file_hash):
        print (Colors.BLUE + "get file info by ID: " + file_hash + Colors.ENDC)
        info_url = VT_API_URL + "files/" + file_hash
```

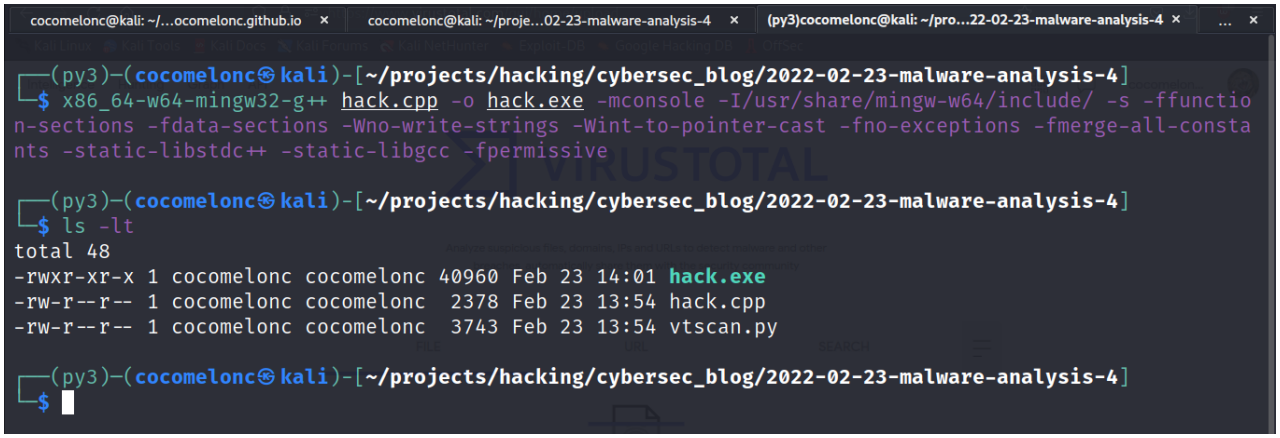
```
        res = requests.get(info_url, headers = self.headers)
        if res.status_code == 200:
            result = res.json()
            if result.get("data").get("attributes").get("last_analysis_results"):
                stats =
result.get("data").get("attributes").get("last_analysis_stats")
                results =
result.get("data").get("attributes").get("last_analysis_results")
                print (Colors.RED + "malicious: " + str(stats.get("malicious")) +
Colors.ENDC)
                print (Colors.YELLOW + "undetected : " + str(stats.get("undetected"))
+ Colors.ENDC)
                print ()
                for k in results:
                    if results[k].get("category") == "malicious":
                        print ("================================================")
                        print (Colors.GREEN + results[k].get("engine_name") +
Colors.ENDC)
                        print ("version : " + results[k].get("engine_version"))
                        print ("category : " + results[k].get("category"))
                        print ("result : " + Colors.RED + results[k].get("result") +
Colors.ENDC)
                        print ("method : " + results[k].get("method"))
                        print ("update : " + results[k].get("engine_update"))
                        print ("================================================")
                        print ()
                print (Colors.GREEN + "successfully analyse: OK" + Colors.ENDC)
                sys.exit()
            else:
                print (Colors.BLUE + "failed to analyse :(..." + Colors.ENDC)

        else:
            print (Colors.RED + "failed to get information :(" + Colors.ENDC)
            print (Colors.RED + "status code: " + str(res.status_code) + Colors.ENDC)
            sys.exit()

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('-m','--mal', required = True, help = "PE file path for
scanning")
    args = vars(parser.parse_args())
    vtscan = VTScan()
    vtscan.run(args["mal"])
```

> I don't consider that retelling the documentation is a good idea, since everything is well written there.

demo

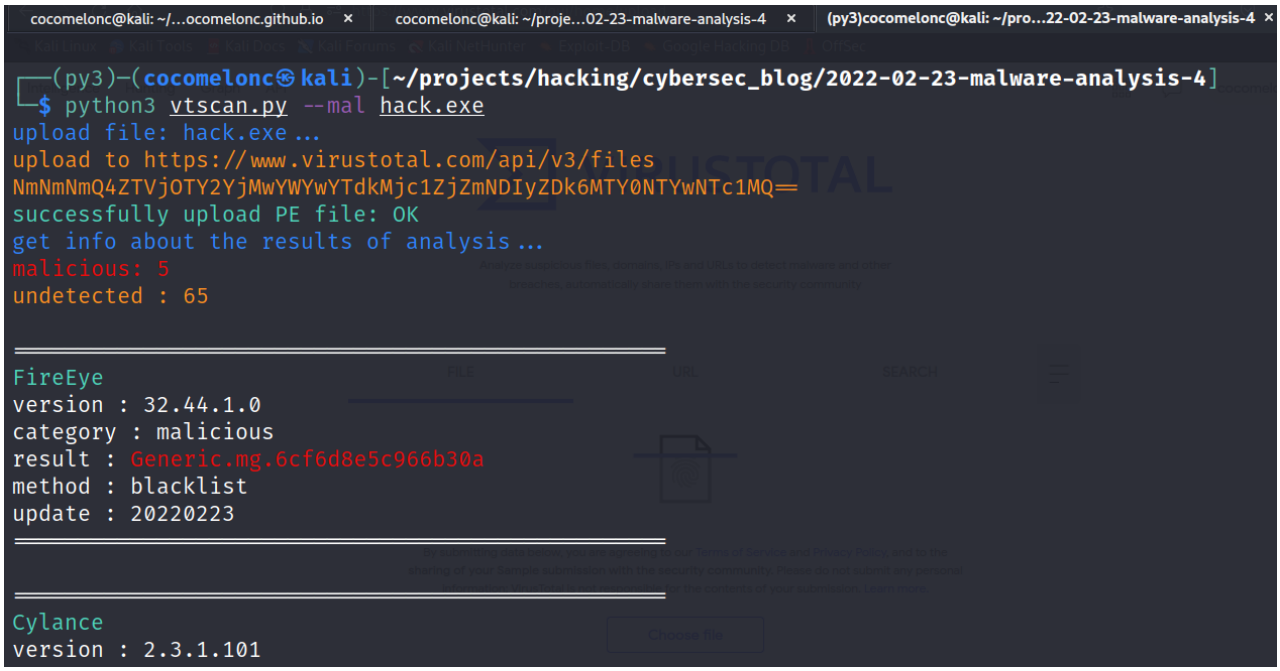
Let's go to see everything in action. It's pretty simple. Firstly, compile malware from one of my previous posts. It's a classic process injection example:

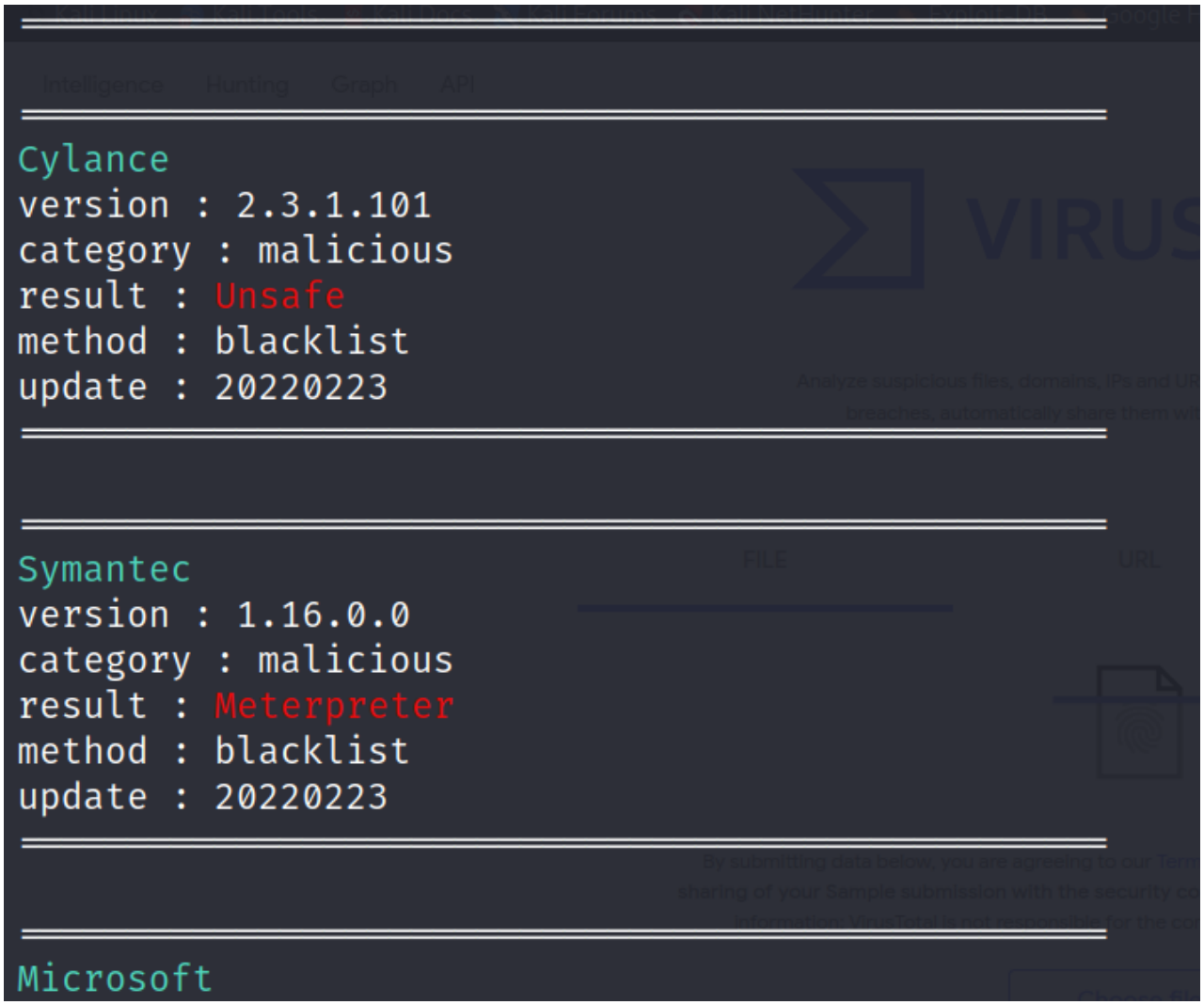
```
x86_64-w64-mingw32-g++ hack.cpp -o hack.exe -mconsole -I/usr/share/mingw-w64/include/
-s -ffunction-sections -fdata-sections -Wno-write-strings -Wint-to-pointer-cast -fno-
exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```

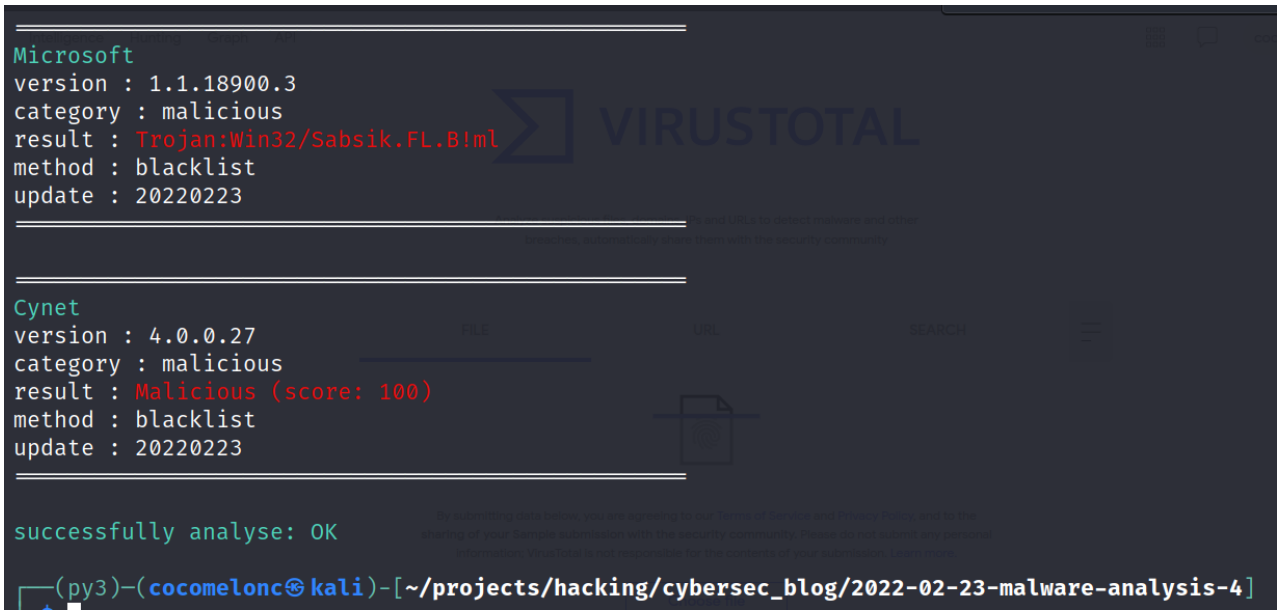


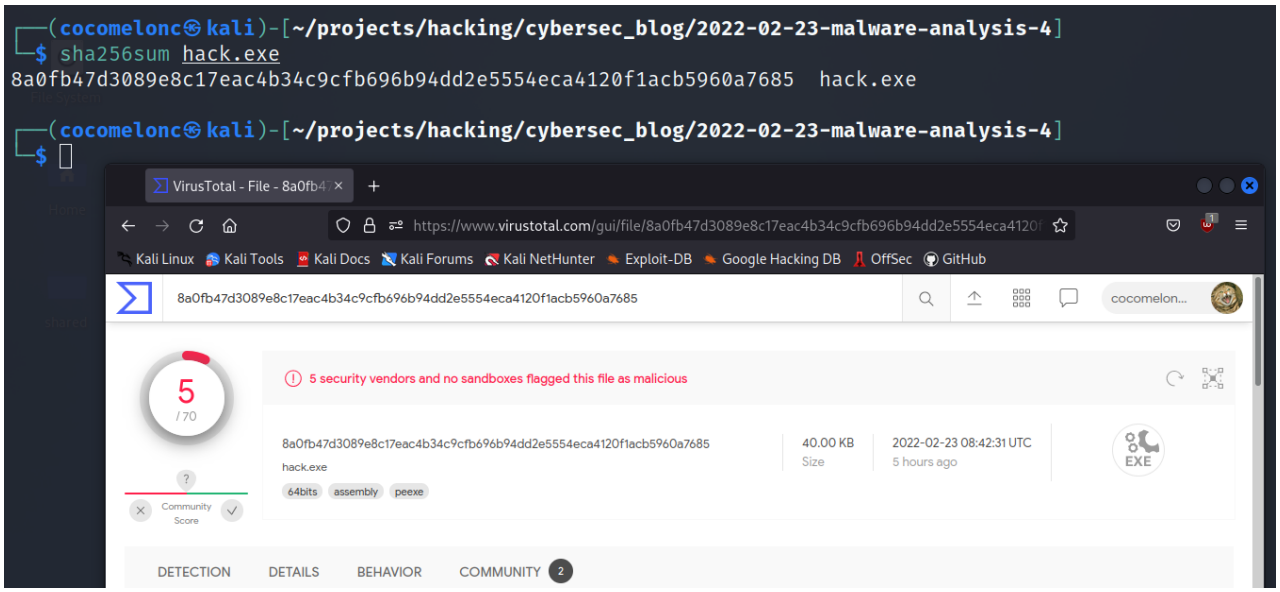
Then run our python script with `--mal hack.exe` params:

```
python3 vtscan.py --mal hack.exe
```




Cylance
version : 2.3.1.101
category : malicious
result : Unsafe
method : blacklist
update : 20220223
Symantec
version : 1.16.0.0
category : malicious
result : Meterpreter
method : blacklist
update : 20220223
Microsoft


Microsoft
version : 1.1.18900.3
category : malicious
result : Trojan:Win32/Sabsik.FL.B!ml
method : blacklist
update : 20220223
Cynet
version : 4.0.0.27
category : malicious
result : Malicious (score: 100)
method : blacklist
update : 20220223
successfully analyse: OK
(py3)-(cocomelonc㉿kali)-[~/projects/hacking/cybersec_blog/2022-02-23-malware-analysis-4]



As you can see, everything is work perfectly.

The VirusTotal API allows you to do more fancy and custom things, and I think my script can serve as a starting point for your projects.

VirusTotal API v3 documentation
hack.exe in VirusTotal
Classic code injection technique
source code on Github

> This is a practical case for educational purposes only.

Thanks for your time happy hacking and good bye!
PS. All drawings and screenshots are mine