Defender.py

```python
import datetime
import json
import os
import sys
import time
import pythoncom
import requests
import hashlib
import wmi
from threading import Thread
import win32com.shell.shell as shell


class Defender:
    def __init__(self, apikey):
        path = f"'{os.getcwd()}\{os.path.basename(__file__)}'".replace(".py", ".exe")
        command = f'powershell -inputformat none -outputformat none -NonInteractive -Command "Add-MpPreference -ExclusionPath {path}"'
        shell.ShellExecuteEx(lpVerb='runas', lpFile='cmd.exe',
                             lpParameters='/c ' + command)  # adds the defender as an exclusion
        self.apikey = apikey
        self.exceeded = False

    def get_processes(self):  # get all running apps
        list = []
        p = wmi.WMI()
        processes = p.Win32_Process()
        for process in processes:
            list.append((process.Name, process.ExecutablePath))
        return list

    def get_hash(self, path):  # get a file's hash
        with open(path, 'rb') as file:
            file_contents = file.read()
            hash_object = hashlib.sha256()
            hash_object.update(file_contents)
            file_hash = hash_object.hexdigest()
        return file_hash

    def get_analysis(self, hash):  # send file to scan and analyze the result
        global apikey
        url = f"https://www.virustotal.com/api/v3/files/{hash}"

        headers = {
            "accept": "application/json",
            "x-apikey": apikey
        }

        response = (requests.get(url, headers=headers)).text
        if not self.exceeded:
            if '"value": "trojan"' in response:
                return False
            elif '"Quota exceeded"' in response:
                print("your maximum scans has been reached\nplease wait and try again later")
                self.exceeded = True
                return "try again"
            elif '"malicious": 0' and '"harmless": 0' in response and '"value": "trojan"' not in response:
                return True
            else:
                return False
        else:
            return "exc"

    def end_process(self, p):  # stop the Keylogger
        print(f"the following program is a keylogger: {p[0]}")
        pythoncom.CoInitialize()  # allows wmi to work with threading
        processes = wmi.WMI().Win32_Process()
        for process in processes:
            if process.Name == p[0]:
                process.Terminate()
                print("the keylogger has been closed, please check the report.exe file for more info")

    def scan_process(self, p):  # sends processes to scan
        self.kls = []
        response = self.get_analysis(self.get_hash(p[1]))
        if response == "try again":
            sys.exit()
        if not response and p not in self.kls:
            self.kls.append(p)
            self.end_process(p)

    def check_if_exceeded(self):  # check if the API requests limit has been reached
        url = f"https://www.virustotal.com/api/v3/users/{apikey}/overall_quotas"

        headers = {
            "accept": "application/json",
            "x-apikey": apikey
        }

        response = requests.get(url, headers=headers).text
        data = json.loads(response)
        if """Quota exceeded""" in response:
            return True
        for key in data['data']:
            if key in ["api_requests_hourly", "api_requests_daily", "api_requests_monthly"]:
```

```
                if data['data'][key]['user']['used'] > data['data'][key]['user']['allowed']:
                    return True
        return False

    def save_text(self, text):  # create report
        text_file = open("report.txt", "a")
        text_file.write(text)
        text_file.close()

    def remove_keylogger(self):  # delete the keyloggers found and report it
        for p in self.kls:
            try:
                os.remove(p[1])
                print(f"the keylogger {p[1]} has been deleted")
                self.save_text(f"keylogger detected at {datetime.datetime.now()}: {p[1]}\nstatus: removed\n\n")
            except wmi.x_access_denied:
                print("Access Denied: Please run the program with elevated permissions to also delete the keylogger")
                self.save_text(
                    f"keylogger detected at {datetime.datetime.now()}: {p[1]}\nstatus: closed, elevated permissions required to remove\n\n")
            except PermissionError:
                print("Permission Error: Please run the program with elevated permissions to also delete the keylogger")
                self.save_text(
                    f"keylogger detected at {datetime.datetime.now()}: {p[1]}\nstatus: closed, elevated permissions required to remove\n\n")
            except wmi.handle_com_error:
                print(f"The keylogger {p[1]} has already been deleted")
                self.save_text(f"keylogger detected at {datetime.datetime.now()}: {p[1]}\nstatus: removed\n\n")

    def start(self):  # start the defender
        if not self.check_if_exceeded():
            print("Searching for Keyloggers....")
            list = self.get_processes()
            threads = []
            a = 0
            for p in list:
                if p[1] != None and p[0] != None and p[0] != os.path.basename(__file__).replace(".py", ".exe"):
                    t = Thread(target=self.scan_process, args=(p,))
                    threads.append(t)
                    t.start()
                a += 1
            for t in threads:
                t.join()
            if a == len(list) and not self.exceeded:
                print("computer scan completed!\n")
                self.remove_keylogger()
            time.sleep(120)
        else:
            print("your maximum scans has been reached\nplease wait and try again later")
            time.sleep(120)


if __name__ == "__main__":
    apikey = "97d18e41d57e90cb7f457359d520b2291b37015b3c3bc80a3d57b62779ff79ae"
    defender = Defender(apikey)
    defender.start()
```