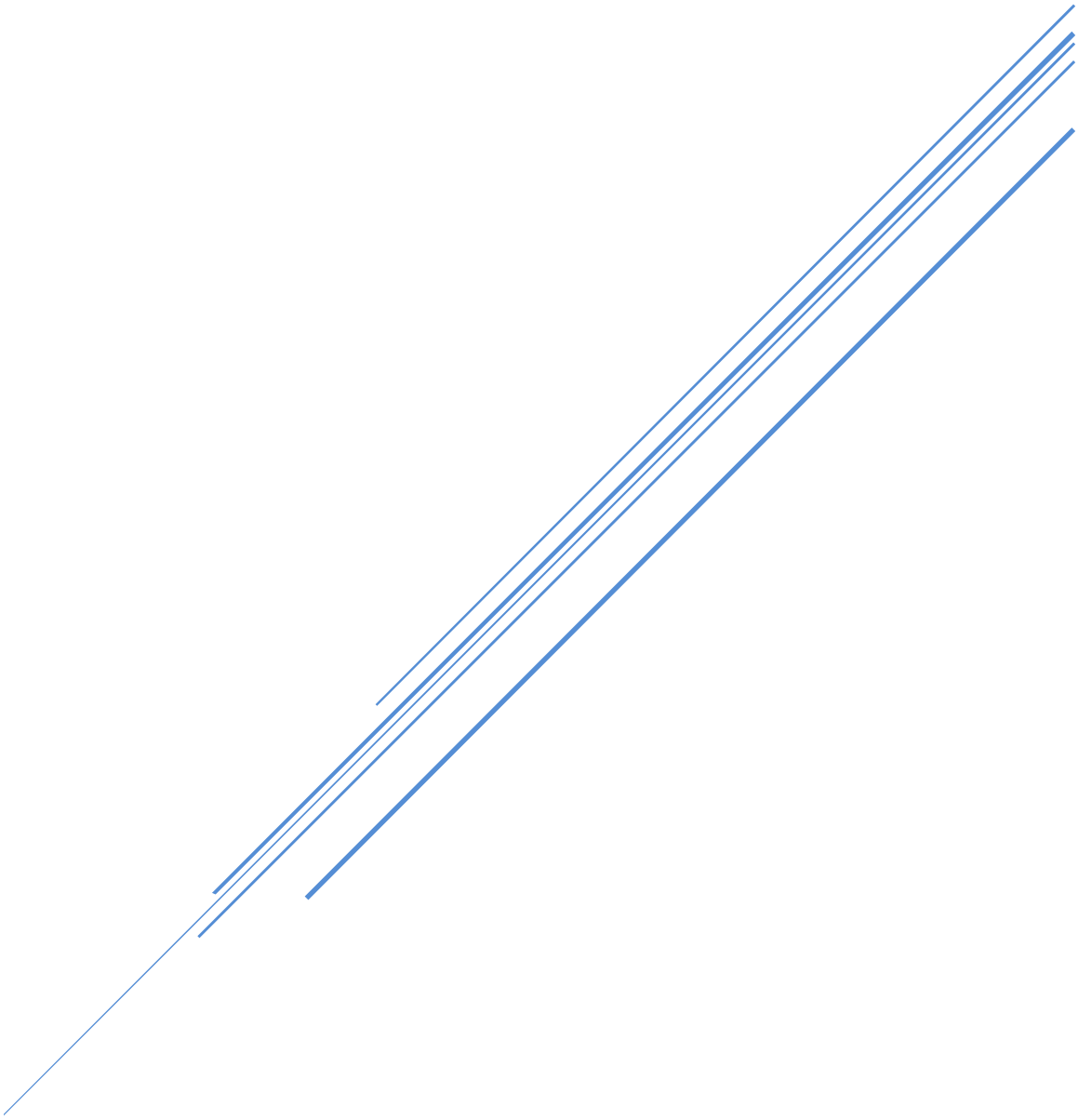
Malware Analysis: Static and Dynamic



Information Security

Table of Contents:

Malware Analysis: Introduction

There are two categories when we are talking about malware analysis:

The first category is **static analysis**, and the second one is dynamic analysis. Static analysis is a method of analyzing a sample or a file at the state it presents itself as without executing the file. So for example if we received an email from someone we don't know that includes a file in the email, what we can do is download the file and not execute it.Now, there are many ways to perform static analysis on a file, something like signature-based detection or permission-based detection, or a source code review.

Dynamic analysis essentially involves executing the sample or the file on your machine and then observing what is going to happen. This is, of course, not safe because this file could be a malware, or could be a ransomware that ends up encrypting all of your files. When we have received an email that includes a file, what we can do with dynamic analysis is to download this file, but this time we are going to execute it. And when we execute it, we are going to monitor how this file actually behaves on our machine. What IP address this file is trying to connect to, or what registry key this file is

trying to create or modify on our machine, or whether this file is trying to download something from the internet.

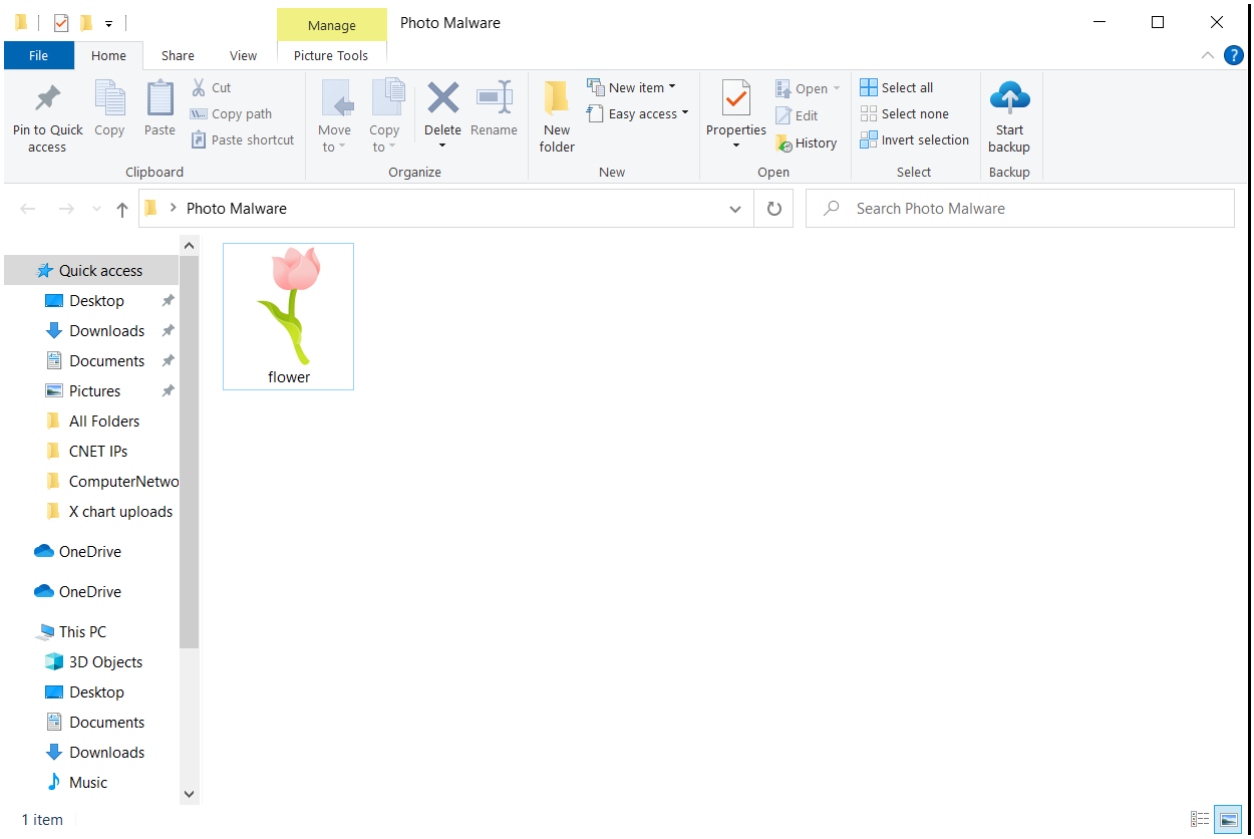
It's not always necessary to download and execute the payload on our machine. Instead, we can use some online services that are called online sandboxing that allow us to upload our file that we think is suspicious to its server and then it will perform the dynamic analysis for us and this is going to be much safer for us. We can see what IP addresses the file is trying to connect to, what registry keys are trying to create or modify, and so on.

Static and Dynamic Malware Analysis using Existing Tools

Static Analysis

File Extension

First we are going to start with static analysis on Windows and then we can move and use static or dynamic analysis on Linux or Windows. Here we have a file that's called **flower**



It has an icon that seems like an image. Instead, it could be something tempting so that you can click on it. As you know that by default, Windows would hide the extension of the files. This is status analysis so you don't actually have to open the file but just for the sake of demonstration, if you open the file,

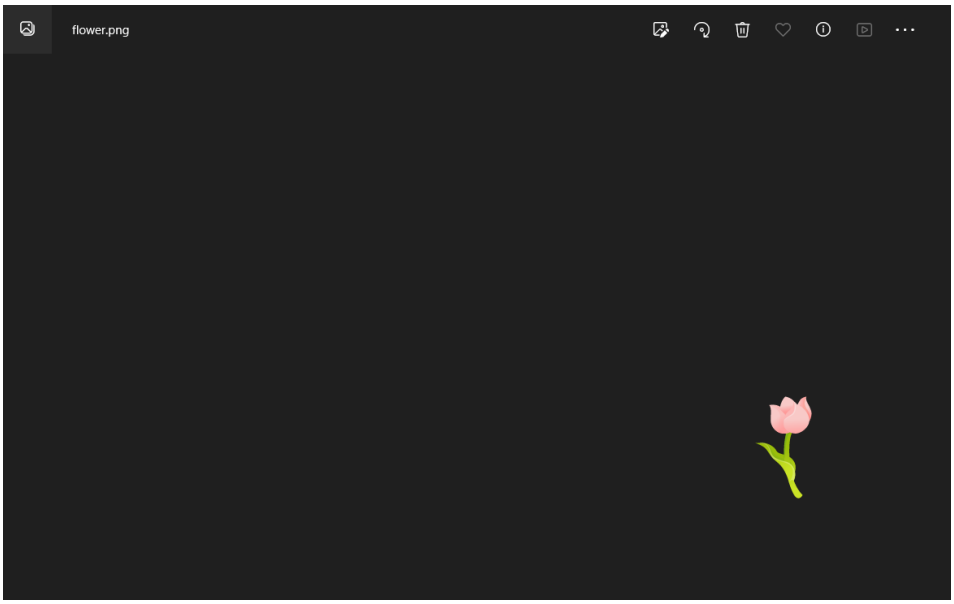
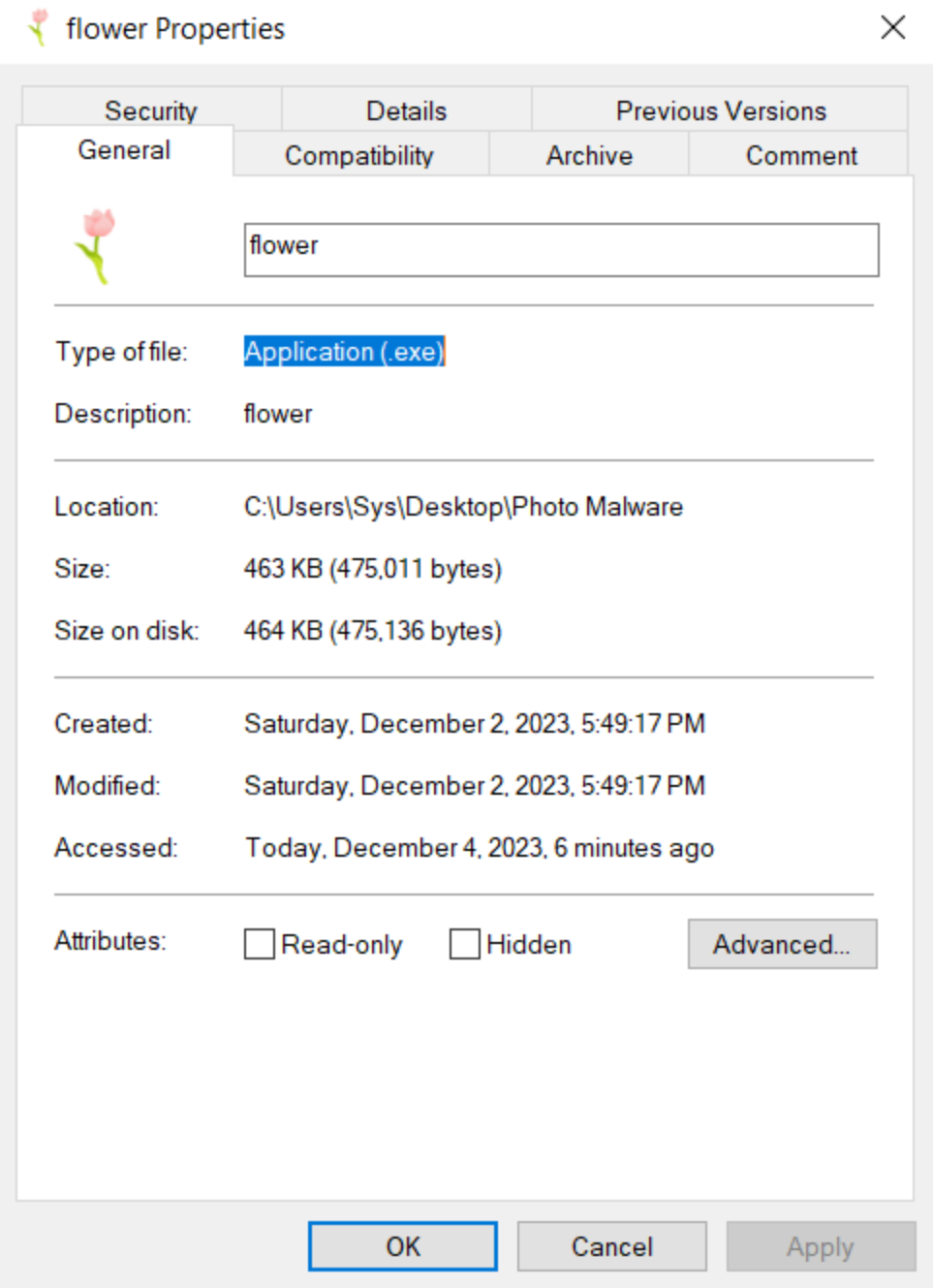


Image of a flower is displayed and at the top left you can see the **png extension.**

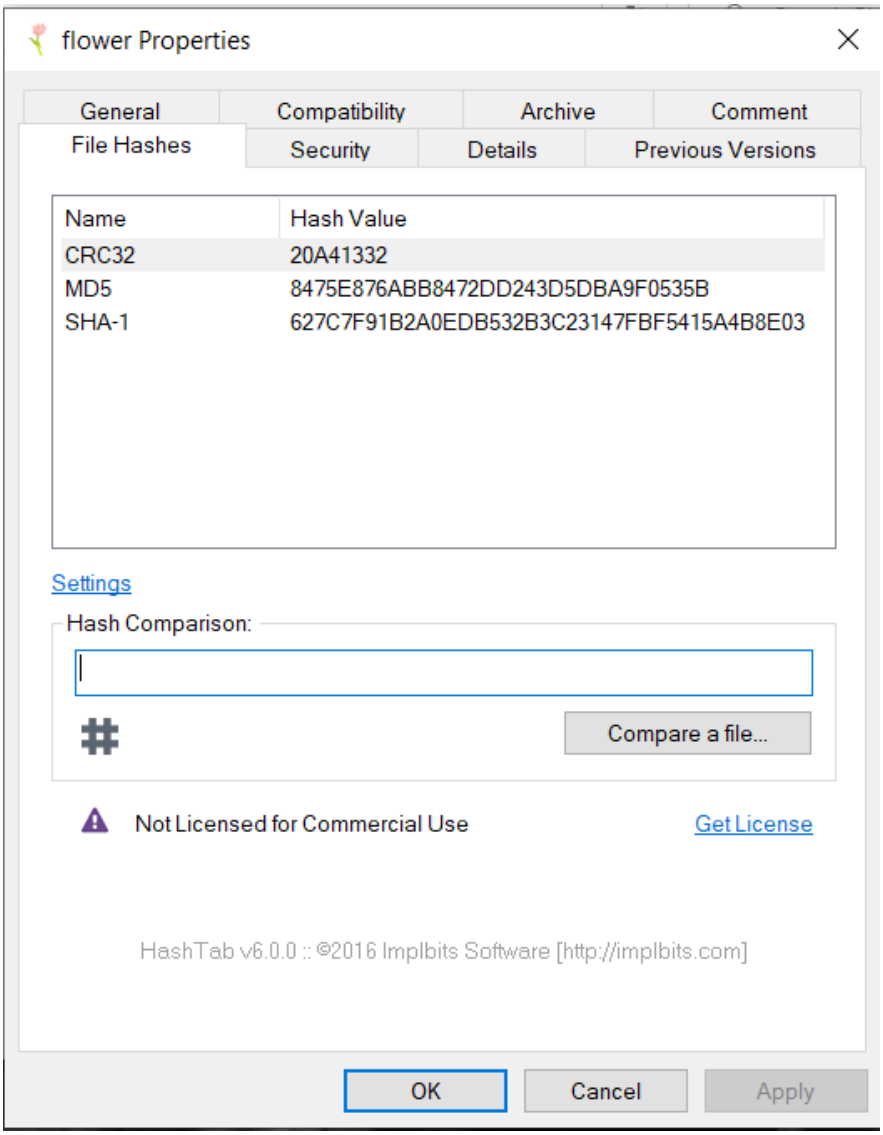
You can either go and view the hidden extensions by changing the settings or you can simply right-click on it and go to Properties, and you can see that the **type of the file is EXE**.



Another thing we can do is to extract the strings that are inside this file. We are going to do this in Linux. I'm not going to do it in Windows because I haven't installed the string tool that will allow me to extract all the strings that are within this executable file or this image.

Extract the Hash

Another thing you can do is that you can extract the hash of the file and then check online whether this is a malicious file or not. I have installed a tool called **Hashtab** When you go to properties, you're going to see the file hashes tab here, and when you click on it, you will get the **md5 hash of this file and the sha1 hash** of this file.

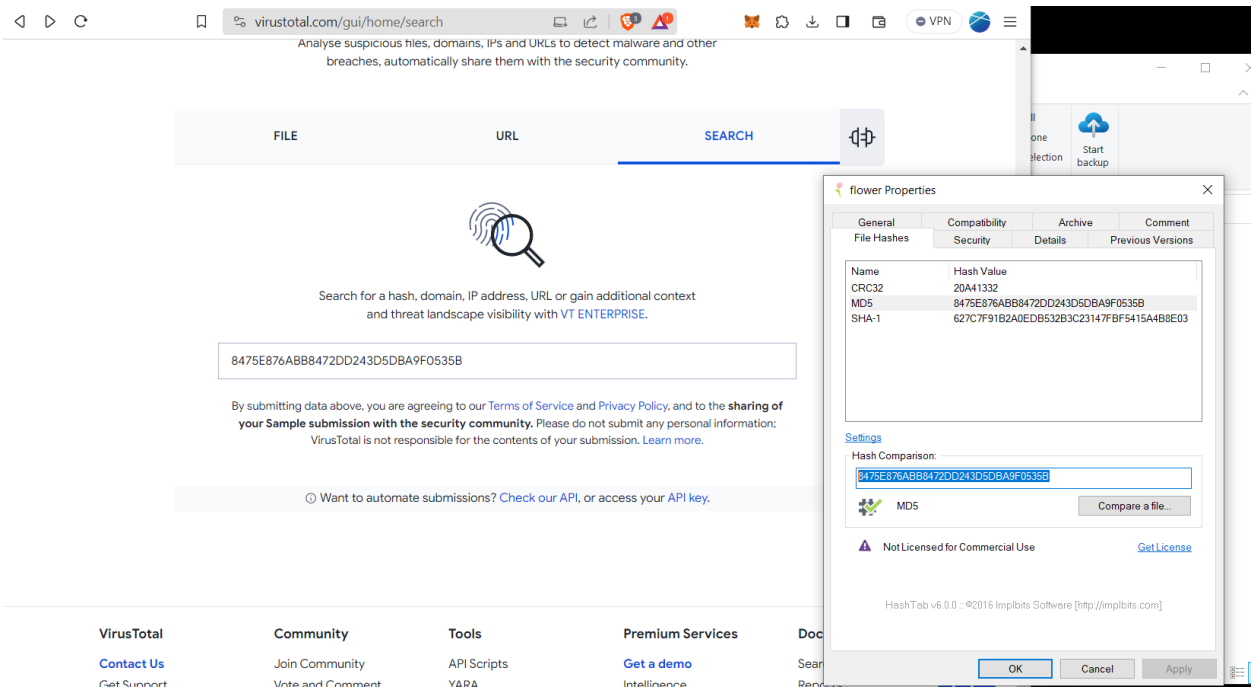


What you can basically do is copy the MD5 hash and then go to a website like Virustotal

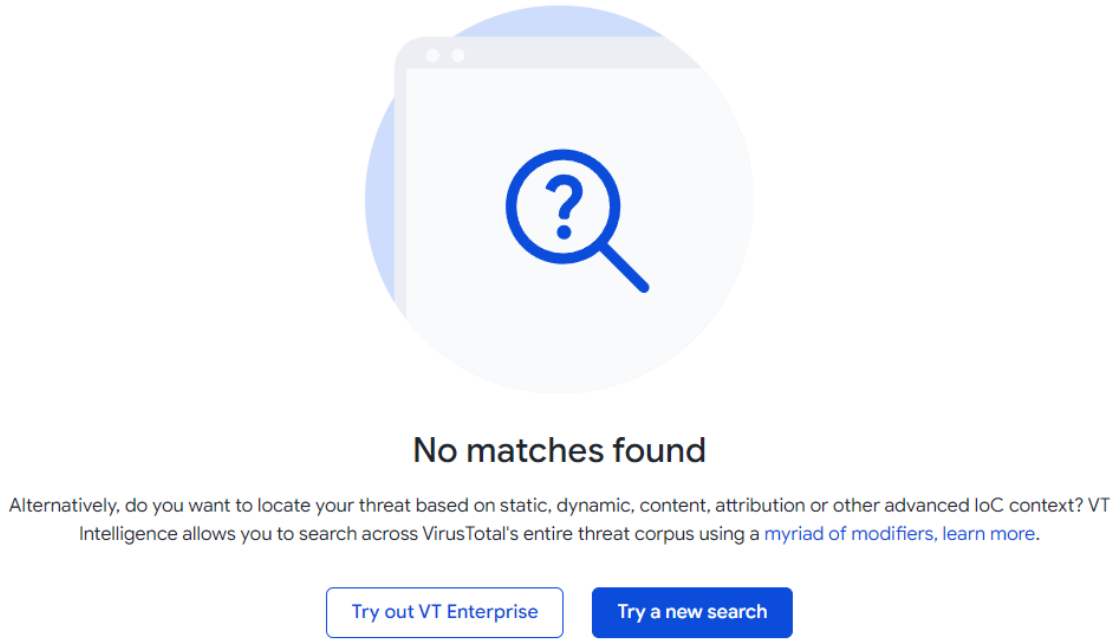
VIRUSTOTAL

Analyse suspicious files, domains, IPs and URLs to detect malware and other breaches, automatically share them with the security community.

go to search and paste the md5 hash.

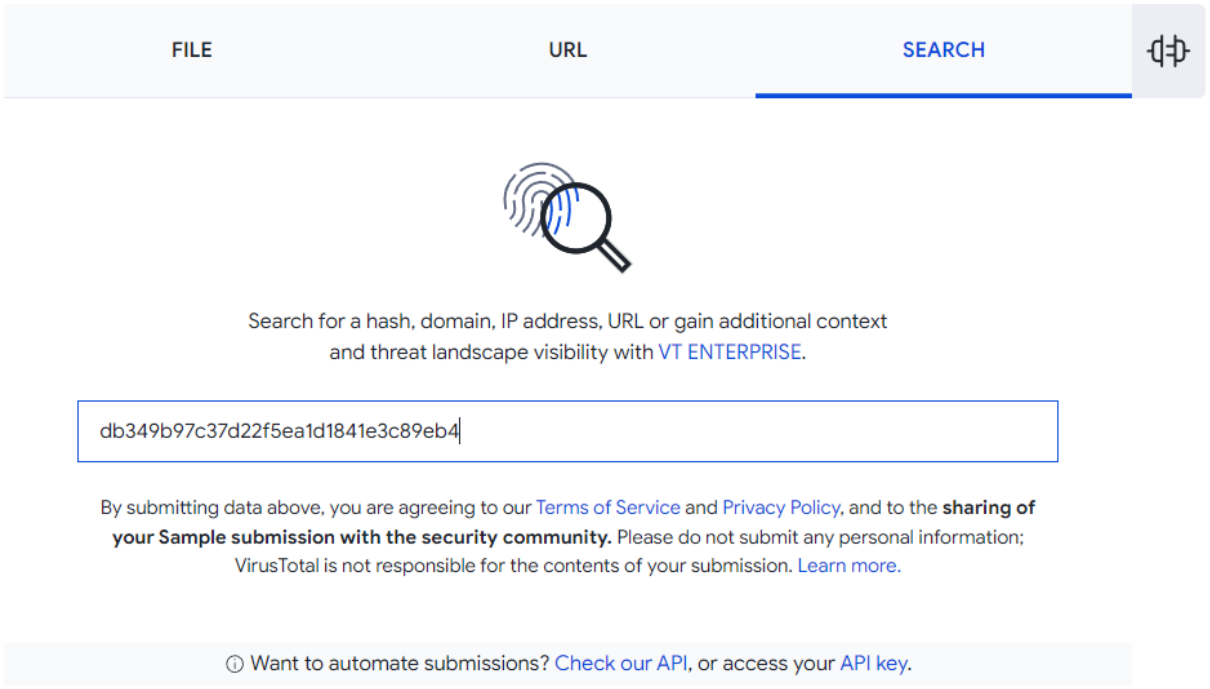


Now, of course, this is only a file that I have created. Nobody scanned this file, that's why it's not suspicious.

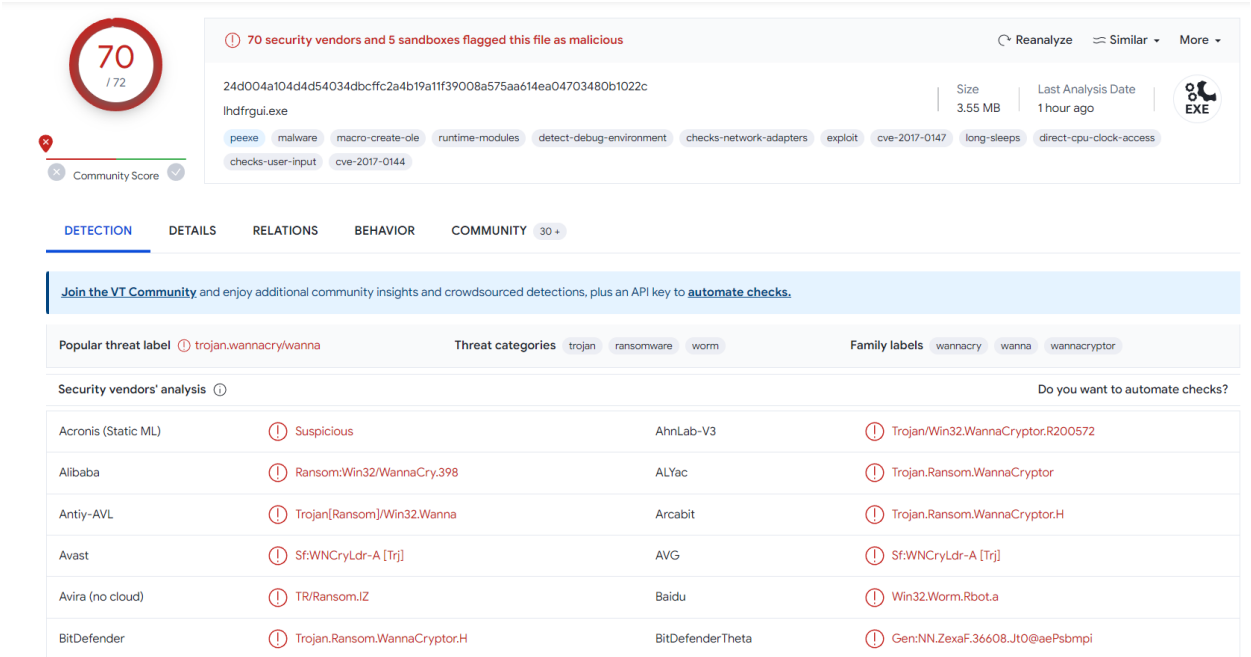


So this is not always a reliable way to check whether this is malicious or not.

But assuming that you have received an email that could be a spam email, and this email has been sent to hundreds, thousands of people, and then people downloaded it and executed it, and then the antivirus reported that it's a malware, then all the results will also be shared with Virustotal. So when you copy its MD5 hash and then put it in VirusTotal, you will get a result because it's a famous malware that's been detected by various antiviruses. For example, when we search for the **WannaCry** hash and we put it in VirusTotal,

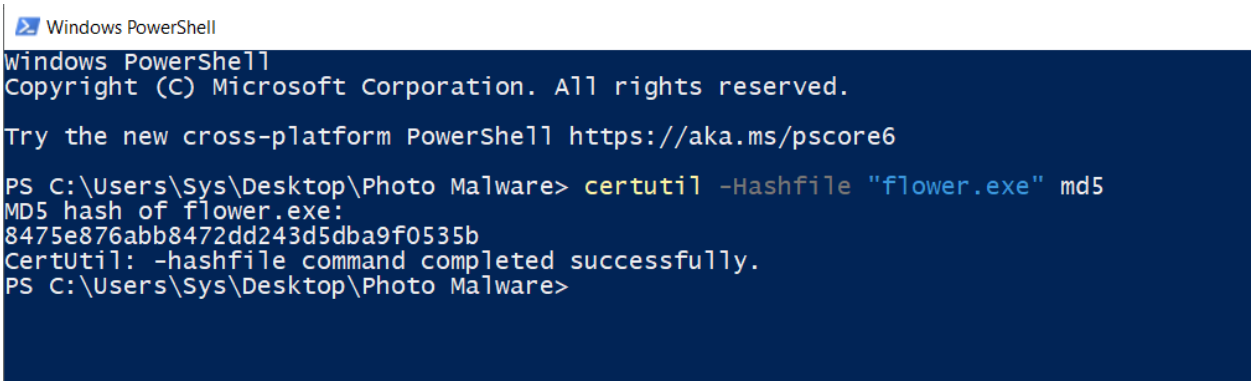


we will see a completely different result. Let me share it for it. This is the service that runs when you execute the WannaCry. When we copy the MD5 hash and then go to virus total and they put it here and then hit Enter, and as you can see



It's been detected by 70 antiviruses out of 72. You can also see some more information like, what it is, its exploit, long-sleeps, macro-create-ole etc, . You can read much more about it and what the antiviruses categorized this file as.

You don't have to use a tool to extract it for you, although it's much easier than using the command line. To do so, I'm going to open powershell in this directory. The comment that we are going to use is: **certutil -Hashfile "flower.exe" md5**. We have to specify the file hash and then the filename, which is called flower.exe, and then the type of the hash we want to extract, whether it's md5, sha1, sha256, etc. I'm going to say md5 and hit Enter. As you can see, this is the MD5 hash of the file without using any tools.



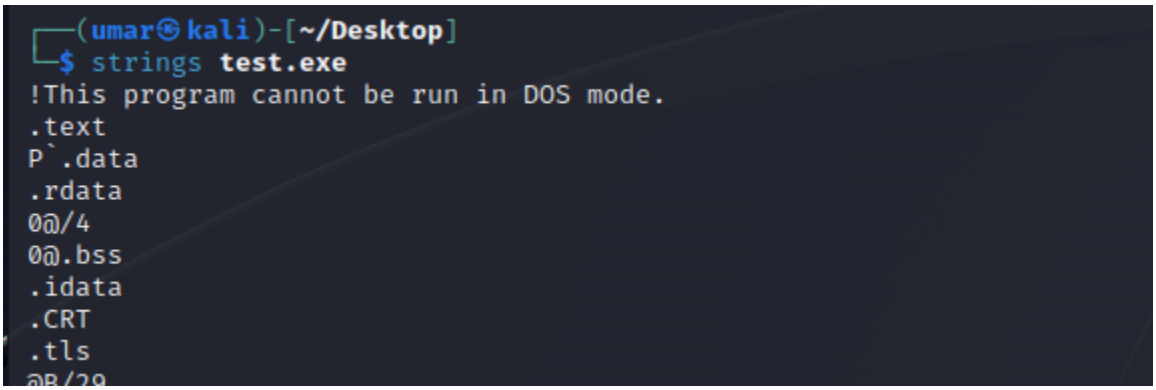
Now on Linux, and what we are going to analyze is this file on the desktop, which is called test.exe or the same flower.exe. Now let's see how we can extract the hash of this file on Linux. So to do so, it's very simple and easy. All we have to do in order for us to extract the MD5 hash is to type in the command **md5sum [filename]**

This is the MD5 hash of the file. Now if you want to extract the shA1 hash, you can just type sha1sum, and then the filename, and here is the hash. You can copy this hash and

put it in VirusTotal to check whether it's malicious or not. You can, of course, upload it if you don't want to extract the hash.

Extracting the strings

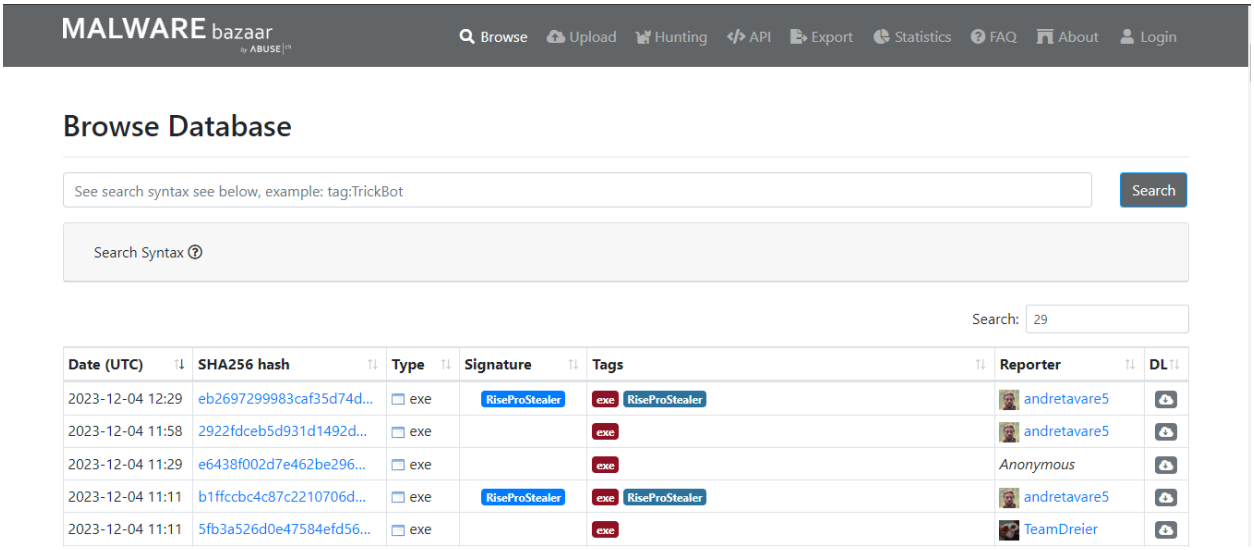
Another command that is useful is by extracting the strings. We can just type strings and then the name of the file. **strings test.exe**



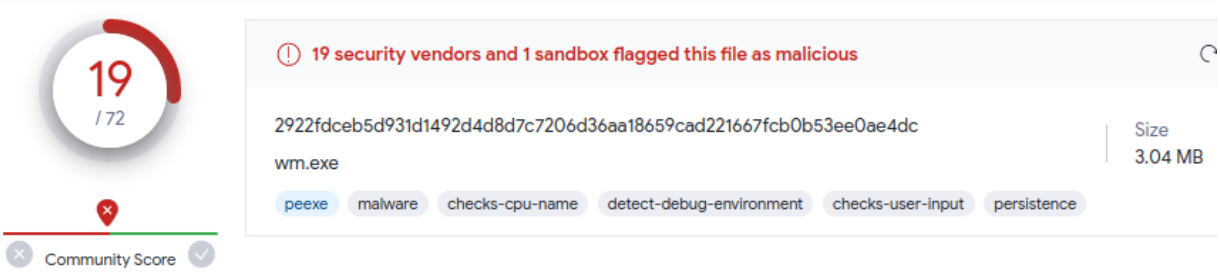
Here are all the strings that we can view in this file. There is a lot of text and if you have noticed anything suspicious, you can search for it online to see if this file is legit or not. The test.exe file I have over here is actually an executable of C++ code I myself write that just creates a simple .txt file

Dynamic Malware Analysis

Now let's assume that we have received a file which is the malware.exe and I know that it looks very suspicious so I'm not going to open it. It is actually a malware I downloaded from a website called **Malware Bazaar** and I renamed it to **malware.exe.** There are other websites like Virus Share, VX Heaven etc where you can download malwares as well. Be sure to download and run them in a controlled environment such as Virtual Box.



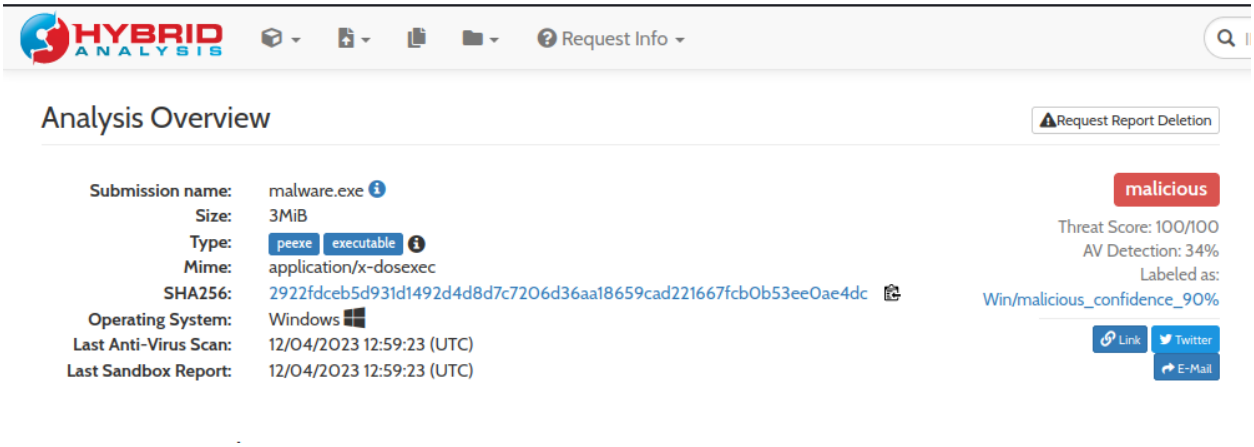
First I did static analysis on it using VirusTotal using its md5 hash and it was deemed as malicious by 19/72 vendors



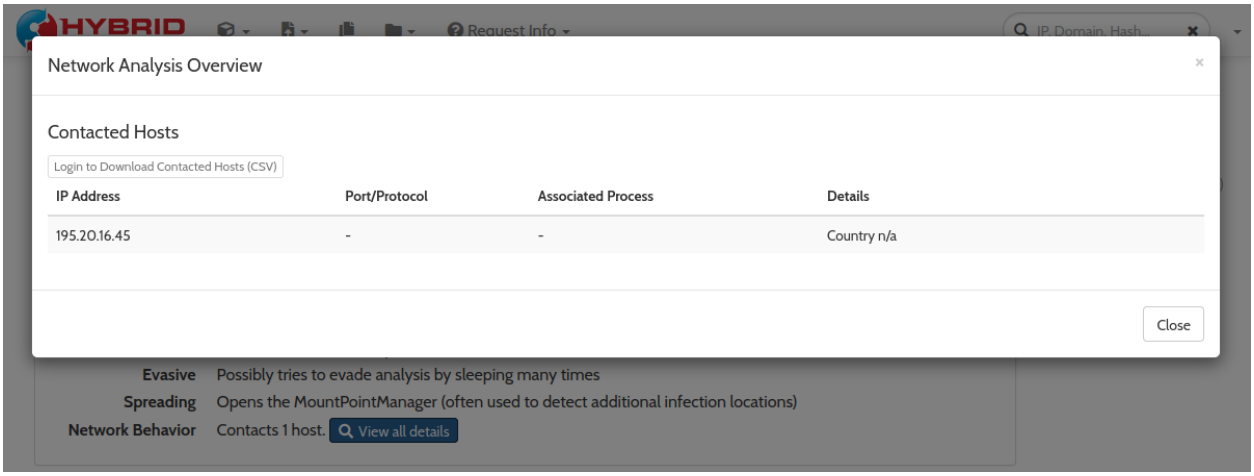
As I said before, dynamic analysis involves executing the file and this is what we are going to do. Although this is dangerous and I do not recommend it if you are not very familiar with malware and malware analysis. Now we have a file that is suspicious. We might copy it or download it in a virtual machine so that we are out of any sort of danger from this file. Another thing we can do instead of executing the payload or the program or the suspicious file on our machine is that we can upload it to online sandboxing services that will execute this file and analyze how it behaves and then it will give us a report. One of the most famous websites are **Any Run** and a website called **Hybrid Analysis**. So here is the first website Hybrid Analysis.

You can just upload your file and then say, Analyze. It will take like 5-10 minutes until it gives you the full report.

After running this file on a Windows 10 Operating System, Hybrid Analysis has deemed this file as Malicious with a **threat score of 100/100**



Furthermore, it also tells us that the malware tried to contact a host with IP address 195.20.16.45

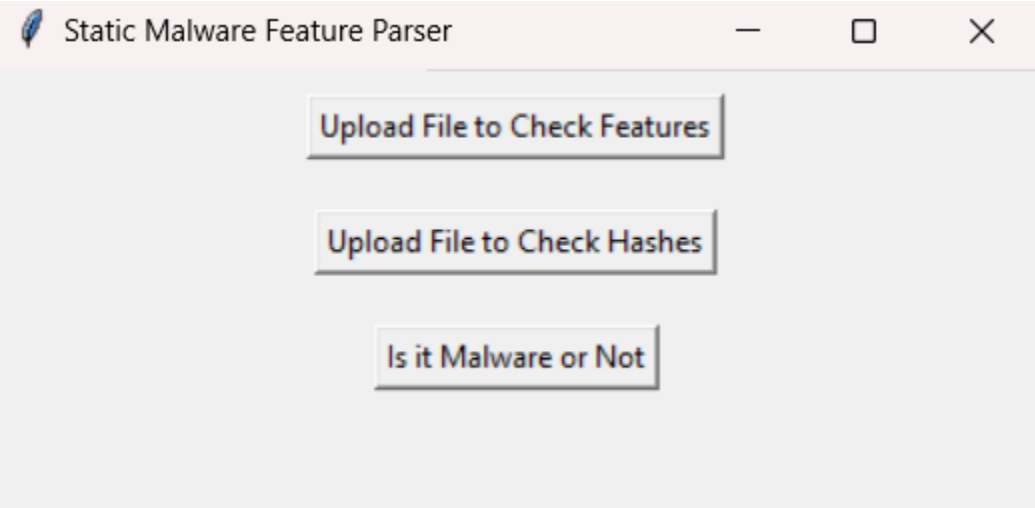


And this is the second website that I think is also very good. You can register on it and then you can use it.

This was a basic introduction on how we can analyze a file that we have received by email or we have got in a USB drive. We have learned the basics of static analysis and dynamic analysis.

Malware Analysis Prototype Development

Static Malware Analysis



Extracting Features from Executable Files: The script extracts various features from executable files (.exe) for analysis. This includes parsing specific attributes of the file using libraries like pefile.

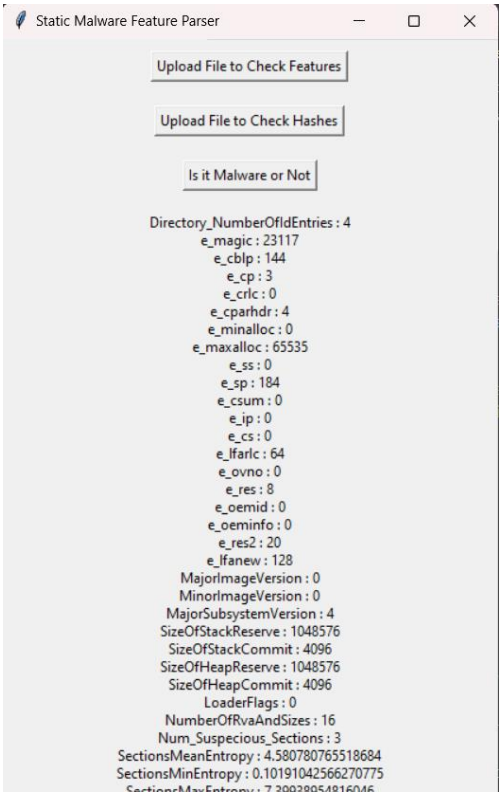
```
import itertools
import pefile
import math
import hashlib

file_path = 'e4dcfe552c8f34aa797aebeb9b68988edb50ebf185e67f0b173ef7e8c57685b0.exe'

def get_entropy(data):
    if not data:
        return 0

    entropy = 0
    for x in range(256):
        p_x = float(data.count(chr(x)))/len(data)
        if p_x > 0:
            entropy += - p_x*math.log(p_x, 2)
    return entropy


def EXTRACT_OPTIONAL_HEADER_INFO(LResource,Resources_header,file_path,fileName):

    pefile_info = pefile.PE(file_path)

    LResource[Resources_header['Name']]=fileName
    LResource[Resources_header['MajorLinkerVersion']] = pefile_info.OPTIONAL_HEADER.MajorLinkerVersion
    LResource[Resources_header['MinorLinkerVersion']] = pefile_info.OPTIONAL_HEADER.MinorLinkerVersion
    LResource[Resources_header['SizeOfCode']] = pefile_info.OPTIONAL_HEADER.SizeOfCode
    LResource[Resources_header['SizeOfInitializedData']] = pefile_info.OPTIONAL_HEADER.SizeOfInitializedData
    LResource[Resources_header['SizeOfUninitializedData']] = pefile_info.OPTIONAL_HEADER.SizeOfUninitializedData
```

Hashing for Malware Identification: It uses hashing techniques (like SHA256) to generate unique identifiers for files, aiding in identifying and comparing malware samples.

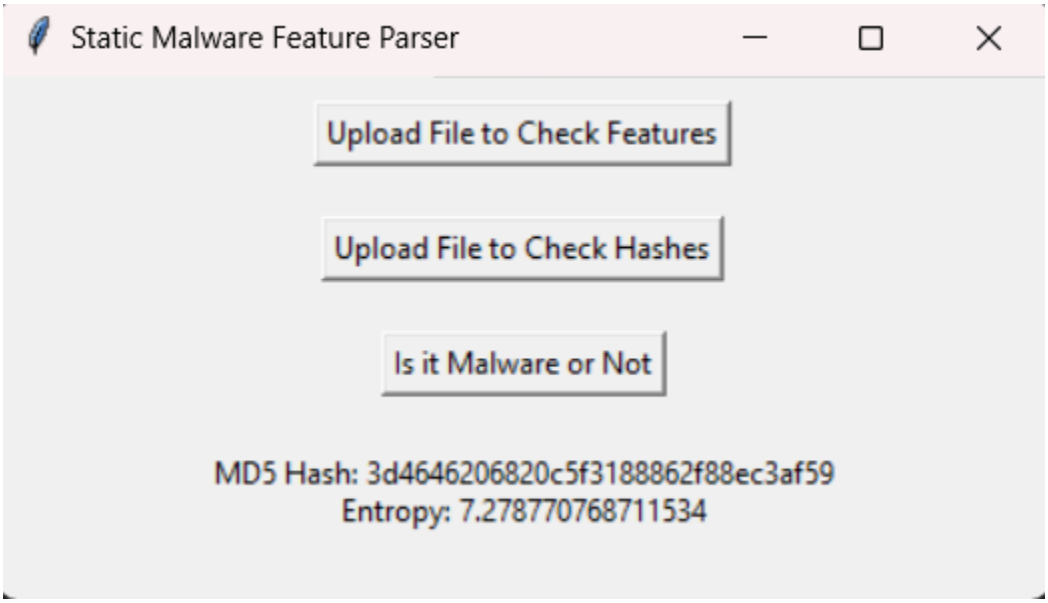
```python
import hashlib

def calculate_hashes(file_path):
    # Create hash objects
    md5_hash = hashlib.md5()
    sha1_hash = hashlib.sha1()
    sha256_hash = hashlib.sha256()

    # Open the file in binary mode and read it in chunks
    with open(file_path, "rb") as file:
        for chunk in iter(lambda: file.read(4096), b""):
            md5_hash.update(chunk)
            sha1_hash.update(chunk)
            sha256_hash.update(chunk)

    # Return the hexadecimal representations of the hashes
    return (
        md5_hash.hexdigest(),
        sha1_hash.hexdigest(),
        sha256_hash.hexdigest()
    )

# Example usage:
file_path = 'e4dcfe552c8f34aa797aebeb9b68988edb50ebf185e67f0b173ef7e8c57685b0.exe'  # Replace with your actual file path
md5_hash, sha1_hash, sha256_hash = calculate_hashes(file_path)

print(f"MD5 Hash of {file_path}: {md5_hash}")
print(f"SHA-1 Hash of {file_path}: {sha1_hash}")
print(f"SHA-256 Hash of {file_path}: {sha256_hash}")
```

```
MD5 Hash of e4dcfe552c8f34aa797aebeb9b68988edb50ebf185e67f0b173ef7e8c57685b0.exe: 3d4646206820c5f3188862f88ec3af59
SHA-1 Hash of e4dcfe552c8f34aa797aebeb9b68988edb50ebf185e67f0b173ef7e8c57685b0.exe: fecc92d2e0bdfdd5abe90548dd5ea3235bc7daeb
SHA-256 Hash of e4dcfe552c8f34aa797aebeb9b68988edb50ebf185e67f0b173ef7e8c57685b0.exe: e4dcfe552c8f34aa797aebeb9b68988edb50ebf185e67f0b173ef7e8c57685b0
```

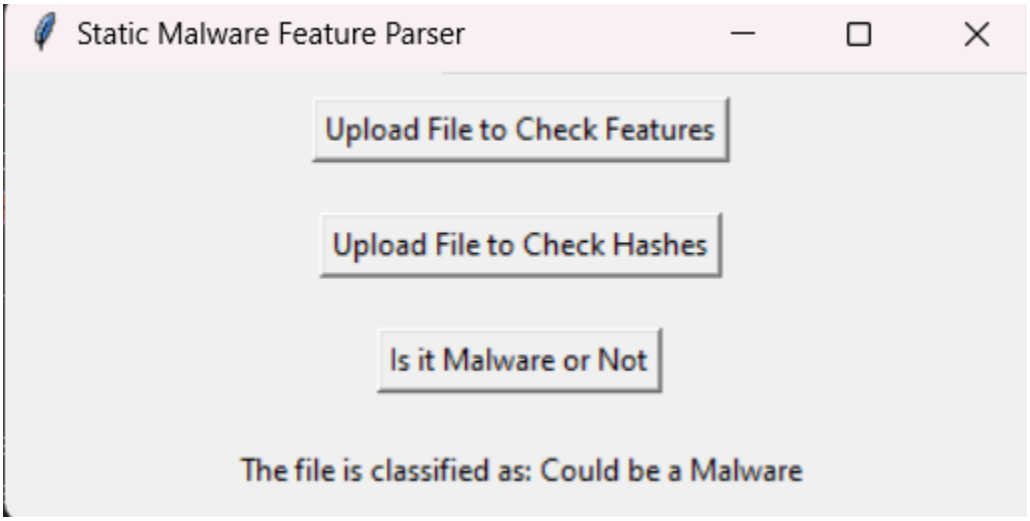
Using External Tools for Analysis: The script likely integrates with external tools or databases (e.g., VirusTotal) for enhanced analysis and comparison with known malware signatures.



GUI for Analysis: The code includes a graphical user interface (GUI) setup, allowing users to upload files for feature extraction and hash checking. This suggests an interactive platform for malware analysis.



Malware Classification: There's functionality for classifying a file as malicious or not, possibly based on the extracted features and comparison against known malware characteristics.

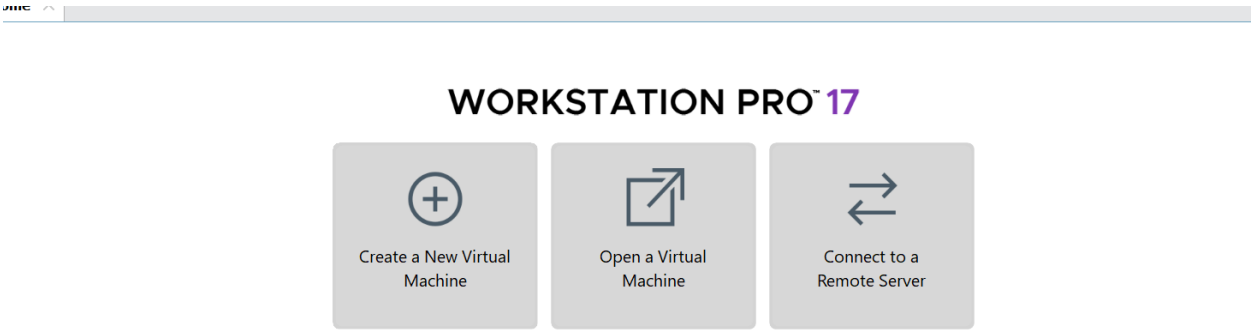


Error Handling and User Feedback: The script handles exceptions and provides feedback to the user, indicating a user-friendly approach to malware analysis.

Main Loop for GUI Interaction: The script runs a main loop for the GUI, allowing continuous user interaction and analysis execution.

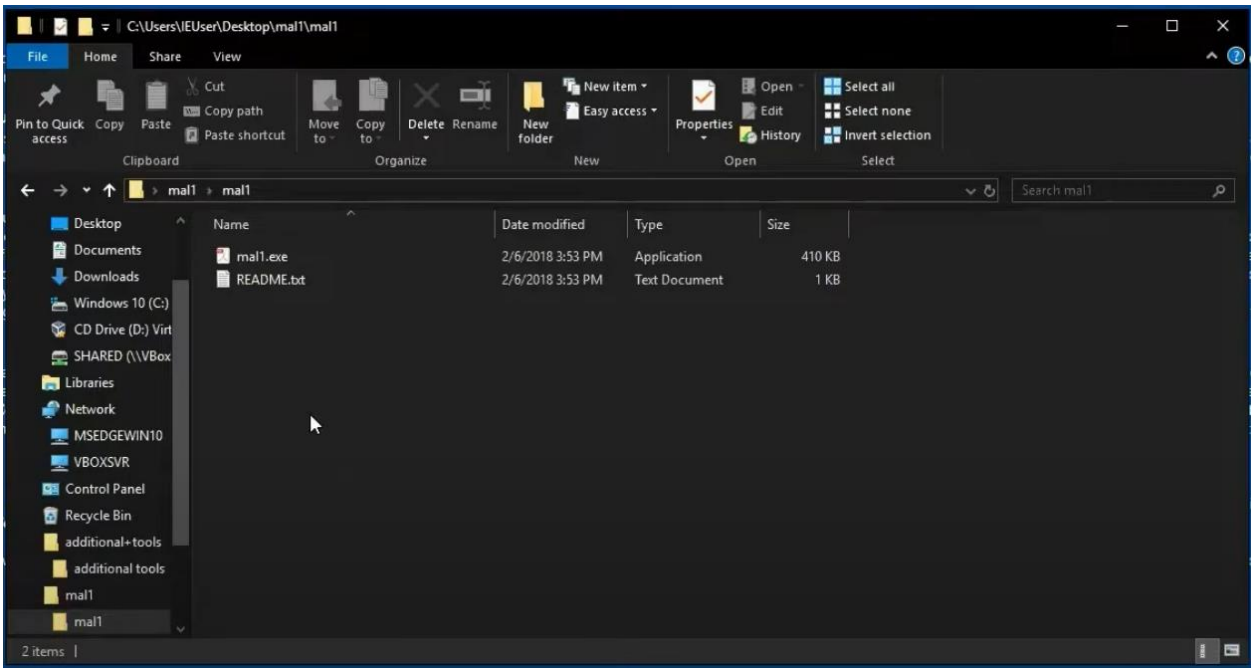
Dynamic Malware Analysis:

In dynamic malware analysis first we will install virtualbox or vmware station then we will install windows 7/10/11.

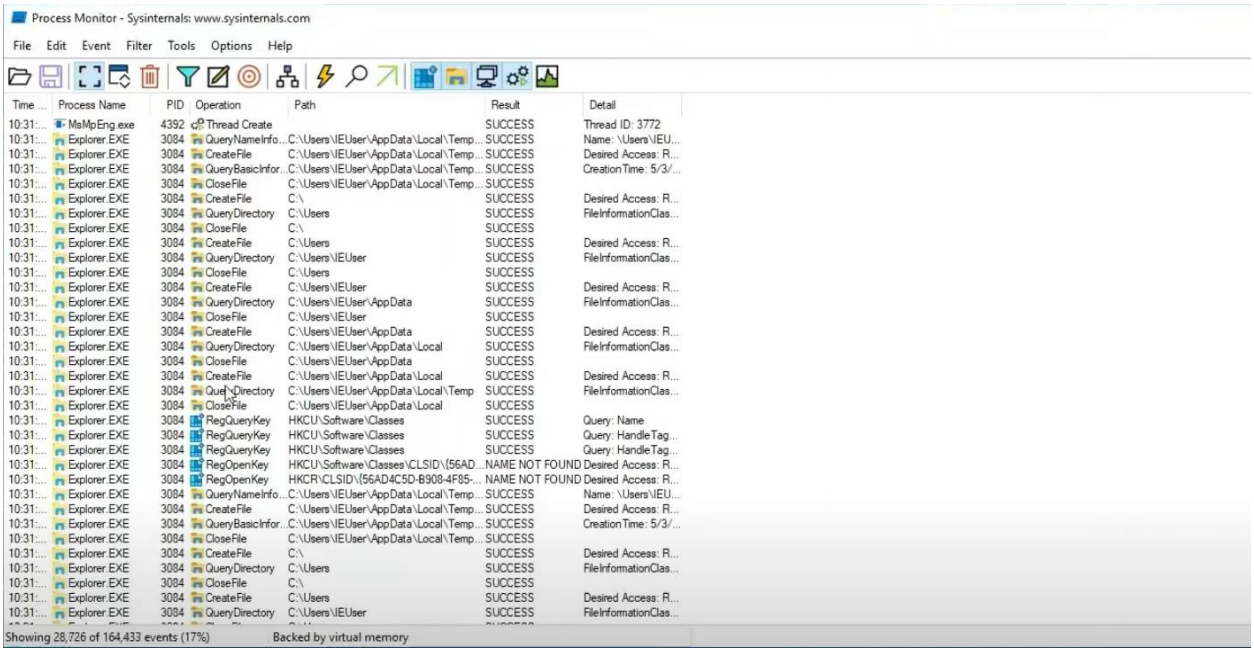


After installing windows 7/10/11 we will open it and then install three different software for our task. Fakenet, procmonitor and then regshot. It will help in our work. First we will download the malware from the website. We will execute it before we need to on the process monitor to monitor the processes, and then fake one in order to show the malware the internet is connected to the system in reality it is not. Then regshot which takes the screenshot of the system. We take regshot before executing it and then after executing then we will compare it and analyze the output.

1. **Malware file**



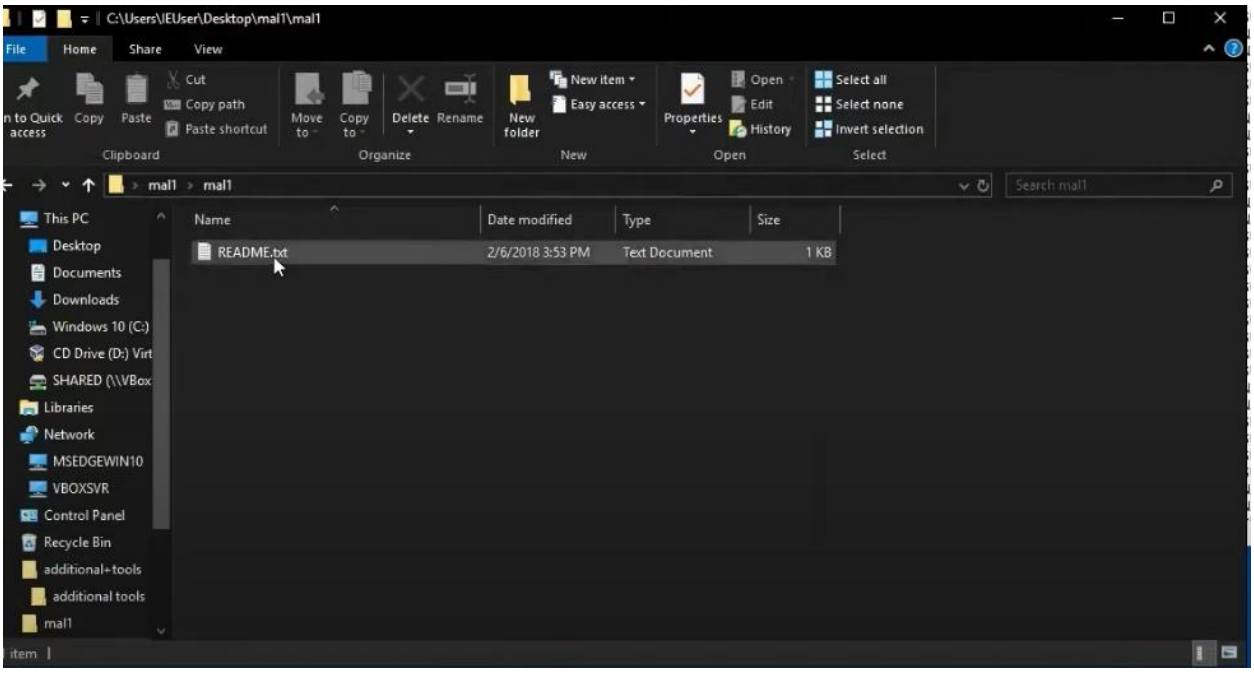
2. Process Monitor



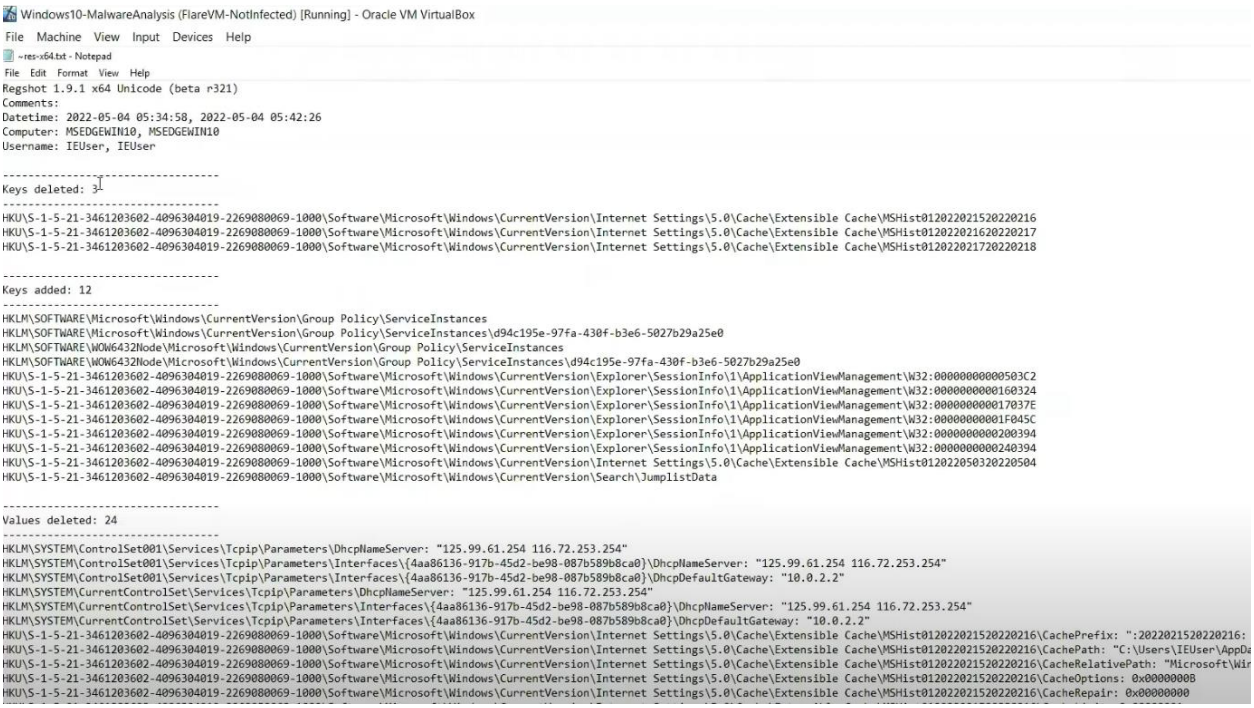
3. Fake net

4. Regshot

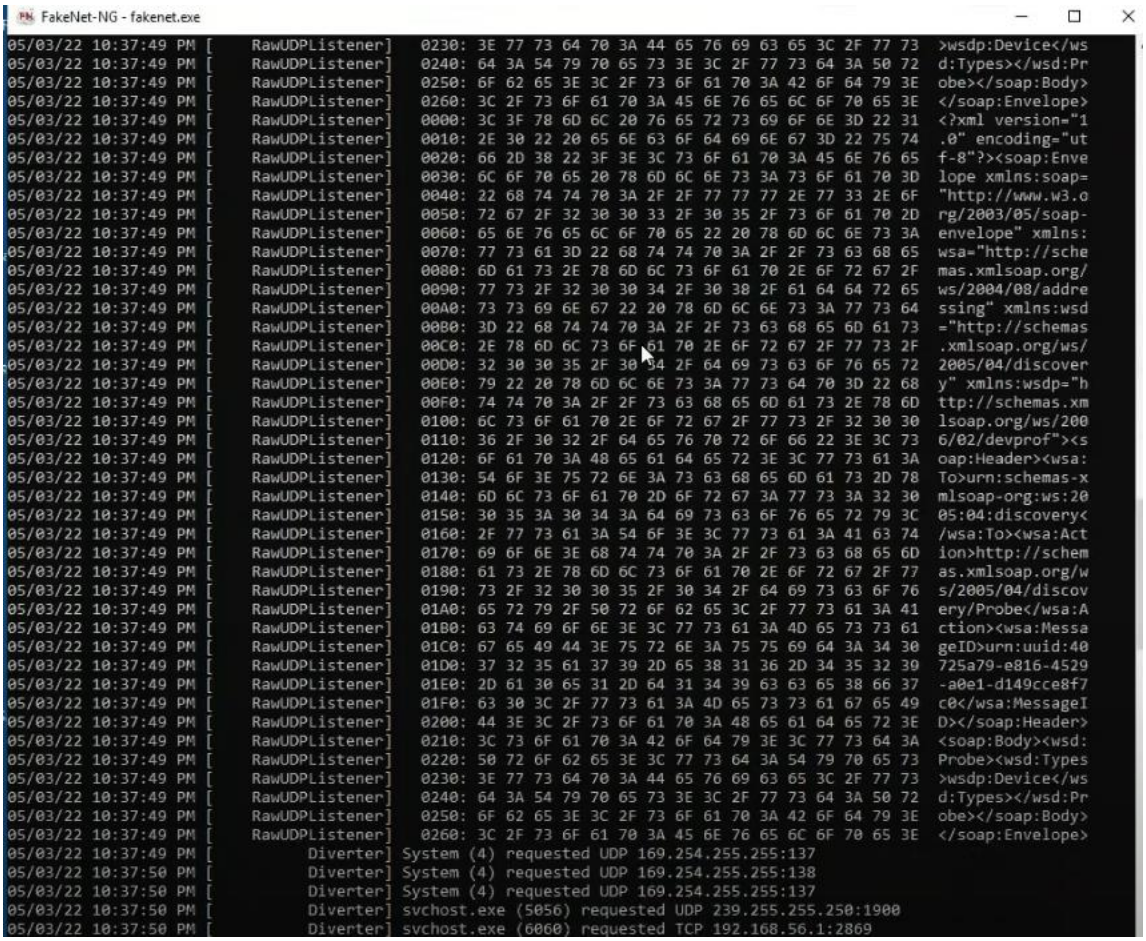
We have taken the screenshot. Now we will execute the malware and after that we will take screenshot again and compare the outputs.



After executing the malware the file has been removed and now we have taken a screenshot again. Now we will compare the first and second shot to compare the out.



if we see in the fake net the malware try to make connection with some ip



Now we have the output after the malware execution. We analyze it using the code.

We have code to analyze it

```python
import json,itertools

def Remove_PC_Name(path_name):
    New_Path_Name=[]
    if 'test' in path_name:
        r_split=path_name.split("\\")
        for strings in r_split:
            if 'test' in strings:
                strings="guest"
            New_Path_Name.append(strings)
        New_Path_Name="\\".join(New_Path_Name)
        return New_Path_Name
    return path_name

def Remove_File_name_Reg(reg):
    if 'RASMANCS' in reg:
        r_split=reg.split("\\")
        new_reg=[]
        for m in r_split:
            if 'RASMANCS' in m:
                m='RASMANCS'
            new_reg.append(m)
        new_reg="\\".join(new_reg)
        return new_reg
    if 'RASAPI32' in reg:
        r_split=reg.split("\\")
```

```python
        r_split=reg.split("\\")
        new_reg=[]
        for m in r_split:
            if 'RASAPI32' in m:
                m='RASAPI32'
            new_reg.append(m)
        new_reg="\\".join(new_reg)
        return new_reg
    return reg

def WriteDictToCSV(dict_data,csv_header,header):

    try:
        with open('Dataset.csv', 'a') as csvfile:
            writer = csv.DictWriter(csvfile, fieldnames=csv_header)
            if int(header)==1:
                writer.writeheader()
            writer.writerow(dict_data)
    except IOError as err:
        errno, strerror = err.args
        print("I/O error({0}): {1}".format(errno, strerror))
    return

def Extract_Registry(Resource,Resources_header, data , file_list ):

    file_name=file_list
    for name in file_name:
```

After executing this we will be able to know what kind of harm malware did to our system.

Literature Review:

- Methodologies and Tools

Static Analysis:

1) **Hashing Techniques:** Used to generate unique checksums for files, aiding in identifying malware variants. SHA256 is a common hashing algorithm used for this purpose.[1]
2) **Fuzzy Hashing:** ssdeep, a fuzzy hashing algorithm, is employed to address hashbusting by creating similarity digests. This method is less sensitive to small changes in a file compared to standard cryptographic hashing.
3) **VirusTotal:** A widely used tool in static analysis, VirusTotal scans malware samples against multiple antivirus engines and reports findings, helping analysts leverage existing knowledge on malware variants.

Dynamic Analysis:

1) **Common Tools:** Tools like IDA Freeware, IDA Pro, x64 debugger, pestudio, OllyDbg, and VMware virtual machines are commonly used for dynamic analysis .[2]
2) **Tool Selection:** The choice of tools depends on the malware's programming language and format. Analysts often choose tools that support multiple formats and focus on making the malware readable for analysis.

---

[1] https://www.techtarget.com/searchsecurity/feature/Top-static-malware-analysis-techniques-for-beginners

[2] https://www.techtarget.com/searchsecurity/feature/Understanding-malware-analysis-and-its-challenges

● Challenges and Limitations

1) **Time and Resource Constraints:** Analysts often face limitations in terms of time and resources, affecting the depth of their research. The choice of malware analysis strategy is influenced by these constraints and the specific questions that need answering.

2) **Mismatch of Expectations:** There is often a gap between stakeholders' expectations and the technical scope of work required for malware analysis. This can result in unrealistic time frames and workloads for analysts.

3) **Lack of Fundamental Computing Knowledge:** Analysts without a strong foundation in computing may struggle to select appropriate tools and fully understand malware samples.

4) **Difficulty in Code Analysis:** Analysts often face challenges in translating low-level assembly language to high-level logic, finding evidence in code, and explaining their findings to non-technical stakeholders.

● Comparative Analysis

Static vs. Dynamic Analysis:

Static analysis allows for collecting data from a suspicious file without execution, providing a preliminary understanding of the malware. It includes techniques like hashing and fuzzy hashing and leverages tools like VirusTotal.
[3]

[3] Imam Riadi, 2015, p.2 https://core.ac.uk/download/pdf/85136527.pdf

Dynamic analysis, on the other hand, involves executing the malware and observing its behavior on the system, using tools like IDA Pro, x64 debugger, and VMware virtual machines.
The choice between static and dynamic analysis often depends on the specific objectives of the analysis, available resources, and the nature of the malware being investigated.[4]

[5]In summary, both static and dynamic malware analysis have distinct methodologies and tools, each with its own set of challenges and limitations. The choice of methodology often depends on the specific goals of the analysis, the nature of the malware, and the resources available. Understanding these aspects is crucial for effective malware analysis.

[4]https://core.ac.uk/download/pdf/85136527.pdf

[5] Imam Riadi, 2015, p6 https://core.ac.uk/download/pdf/85136527.pdf