MALWARE SCANNER INCORPORATING MD5 HASH-MATCHING USING STATIC AND DYNAMIC DATABASES

A project report submitted in fulfilment of the requirements for the Major Project in

MALWARE ANALYSIS (IT-321)

Submitted by

Anubhav Sharma and Arjun Choudhry
{2K18/IT/029, 2K18/IT/031}

on

16th November 2020

Under the Guidance of

Dr. Kapil Sharma
Head of Department
Department of Information Technology
Delhi Technological University, New Delhi-110042

DELHI TECHNOLOGICAL UNIVERSITY (FORMERLY DELHI COLLEGE OF ENGINEERING)

ACKNOWLEDGEMENTS

The successful completion of this project wouldn't have been possible without the help and support of many, and we would like to extend our sincere gratitude to them.

We are highly indebted to Dr. Kapil Sharma for being a source of constant motivation, and for helping us improve the project to achieve its present shape. We would also like to thank him for giving us valuable insights on report writing and referring research papers.

It would be unfair, should we miss out on thanking our fellow classmates, for some wonderfully insightful discussions on Malware Analysis techniques, for working around us, and for making the subject even more interesting and for sharing many unique ideas, both on and off the subject.

Last but not the least we would like to thank our parents for always being there, and guiding us through this journey.

CONTENTS

ABSTRACT

It is clear that with the advent of internet, it has become practically impossible to not have personal data over the World Wide Web. With the exponential increase in data, the threat to its security has also risen exponentially. Hence the use of tools for the protection of the same have become de-facto mandatory. The threat due to malware cannot be overemphasized.

One of the basic principles that can help protect against malwares is **Hash Matching**. In general, matching hashes for ensuring a file isn't malicious is used in the two following ways:

1. Matching the hash code of the downloadable file with the one published by the organization publishing the file: This is a common technique used by common file hosts like Oxygen Updater, apkmirror.com, androidfilehost.com, etc, to ensure that the files are malware-free. The hash codes of the files are displayed in the download window, which users can also compare with the hash codes published by the app developer. If the hash codes are same, the files haven't been tampered with, and are safe to download.

2. Checking the hash codes of a file against a database of malicious of hash codes: This is a common technique used by Anti-virus software in their preliminary checks, to compare the hash codes of files against a large database of malicious hashes, to identify common malwares, if present in a system.

We intend to build an application primarily based on the latter principle. Also, extraction and checking of specific **strings** in a file may hint towards it being malicious. We intend to also use this ploy to provide a highly accurate and fast malware scanning software.

Here, we have made an application, complete with its own GUI, to be able to quickly detect malwares in a particular file or directory.

1 INTRODUCTION

Any software that is intentionally designed to cause damage to a computer, client, server, or a computer network is a Malware or a Malicious Software. In the present day, a wide variety of malwares exist, ranging from computer viruses, worms, Trojan horses and wiper, to spyware, adware, ransomware, scareware, and rogue software. They typically consist of code developed by cyber-attackers, designed to cause extensive damage to systems and data, or gain unauthorized access to a computer or a network.

According to **McAfee Labs Threats Report[1] for November 2020**, the number of new malware detected in Q2 2020 crossed the 50 million mark for the third time in four quarters, taking the total different known and recognised malware tally up to over 1.2 billion.

A common technique used nowadays to detect malicious files in a computer, or to ensure that a file hasn't been tampered with, is by comparing the hash codes of the original file with the copy generated by the user. If the hash codes match, the file is authentic. At the same time, the same technique is also used for detecting common malwares in a system, whose hash codes are stored in the anti-virus software's database.

Hashing, also known as fingerprinting, is commonly used to produce a unique hash code, which is used to identify the file. The **Message-Digest Algorithm 5 (MD5)[2][3]** hash function is the hashing algorithm most commonly used nowadays, although various versions of the **Secure Hash Algorithm (SHA)** are also very popular.

Thus, in our project, we use the MD5 hash function to generate unique hash codes for a file or directory, and then compare the hash code to 2 datasets, i.e. a static dataset downloaded from **VirusShare[4]** by the program on to the user's computer, and a dynamic dataset by **VirusTotal[5][6]**, available online and accessed using a unique API key.

2 METHODOLOGY

2.1 Datasets and APIs

In our project, we compare the generated MD5 hash code to 2 datasets, i.e. a static dataset downloaded from **VirusShare** by the program on to the user's computer, and a dynamic dataset by **VirusTotal**, available online and accessed using a unique API key.

2.1.1 Static Dataset

The static dataset, made available by **VirusShare**, is downloaded using the 'Update Database' button in the GUI of the program, using the 'curl[7]' library in C++, and the newly extracted database replaces the previously stored database on the user's system. The latest iteration of this dataset, at present, contains 100 text files, each consisting of over 140,000 MD5 hash codes of malicious files, giving us a dataset of over 14 million hashes of malicious software. To check whether a file or directory is malicious using the static dataset, the user needs to use the 'Start Local Scan' button in the GUI of the program. This doesn't require an internet connection, as the database is stored locally.

2.1.2 Dynamic Dataset

The dynamic dataset is available online by **VirusTotal**, and can be accessed using their API and a unique API access key. We call this dataset dynamic, as the dataset is updated to include the latest malicious hash codes without the user's knowledge. To check whether a file or directory is malicious using the dynamic dataset, the user needs to use the 'Start Internet Scan' button in the GUI of the program, and should have a stable internet connection.

2.2 Hash Function

In our project, we use the **QCryptographicHash[8] library** of C++ to extract the **MD5** hash code of a file. The QCryptographicHash class provides a way to generate cryptographic hashes. The enum 'QCryptographicHash::Algorithm' consists of multiple hash functions, with the value '1' indicating MD5 hash function.

2.3 Implementation Environment

Programming Languages Used	C++
Operating System	Created in Linux, but compatible with both Windows and Linux
Library, Packages or APIs Used	Qmake Build QT Libraries QCryptographicHash library curl library VirusTotal API
Interface Design (GUI)	Created using QT Framework
Database Packages Used	VirusShare Hashes Database

Table 1: Details of the implementation environment of the program, to ensure stability and compatibility across systems and operating systems.

3 MALWARE SCANNING METHODS

3.1 Local Scan

The local scan method uses the static database stored on the user's computer to check whether a file is malicious or not. After the user selects the file and local scan option, the file destination is extracted, and the corresponding hash code is generated using the QCryptographicHash library. The database is accessed, and a loop is run till the hash code matches with a value in the database or the loop condition is not met.
If the hash value matches with a value in the database, the file is malicious, and the action frame in our GUI is activated, which displays the option to delete the file directly from our application.

3.2 Internet Scan

The internet scan method uses the dynamic database stored in VirusTotal's server, and is accessed using the VirusTotal API to check whether a file is malicious or not. After the user selects the file and internet scan option, the file destination is extracted, and the file is uploaded to VirusTotal server using the POST function of the API.
A system call is executed to generate and extract the analysis token from VirusTotal API using the curl library. The analysis token is piped into a temporary file, and read from there. The analysis token in the body of the API call is used to get the results. The system call is executed again, and the results are piped into another temporary file, which is parsed to check if the file is malicious.
The contents of the temporary file are loaded into the Details window of the GUI, if the user clicks on the 'Examine Strings' button. If the file is malicious, the action frame in our GUI is activated, which displays the option to delete the file directly from our application.

3.3 Update Database

The local database can be updated by the users, should there be any improvements made by the VirusShare team. The user needs to click on the 'Update Database' button in the GUI, which runs a loop over the number of files available on the VirusShare website.
A URL is created in the format required to download the file using the curl library. The URL is parsed to the curl-based system call, which downloads the file. The system call is executed, and the results are piped to the application database. The existing files are automatically replaced.

4 ALGORITHMS FOR SCANNING METHODS

Algorithm 1 Local Scan

Input:
F_{path} = Path of the file;

Output:
S : Number of files in dataset
0 : N ← 0
1 : hash ← GenerateHash(F_{path})
2 : while N < S do
3 : n_N ← Number of lines of hashes in Nth file
4 : L ← 0
5 : while L < n_N do
6 : if HashAt(L) = hash
7 : ActionFrame(Malicious)
8 : end Program
9 : end if
10 : L ← L + 1
11 : end while
12 : N ← N + 1
13 : end while
14 : ActionFrame(Safe)
15 : return

GenerateHash(): Uses the QCryptographicHash library to generate the MD5 hash code of the file
ActionFrame(): Displays the status of the file, i.e. Malicious or Safe.
HashAt(): Gets the hash value at the given line of the file.

Algorithm 2 Internet Scan

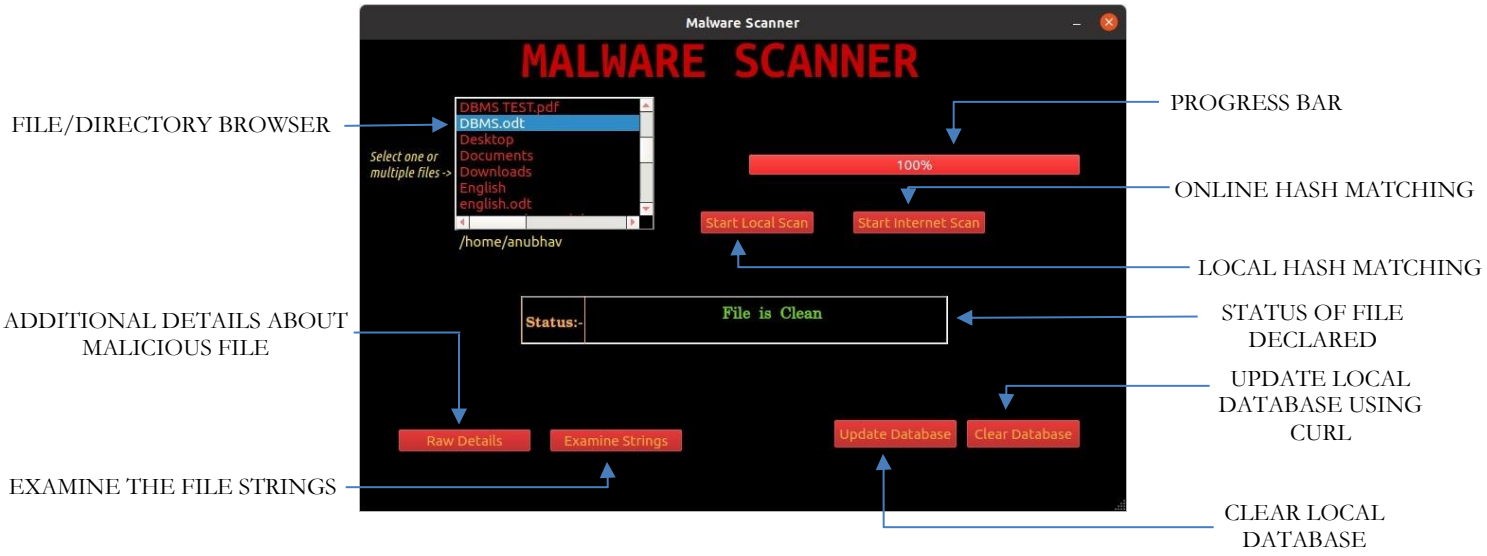
Input:
F_{path} = Path of the file;

Output:

```
1 :  Acmd ← CurlPost(URL of the VirusTotal API file upload, Fpath)
2 :  Acmd ← Acmd + "> temp.txt "
3 :  System(Acmd)
4 :  DetailToken ← Read (temp.txt)
5 :  Gcmd ← CurlGet(DetailToken)
6 :  Gcmd ← Gcmd +"> Results.txt"
7 :  System (Gcmd)
8 :  Lin ← 619
9 :  Textstr ← TextStream(Results.txt)
10 : while Lin--  && TextStr
11 :       Cline ← Read(TextStr)
12 : end while
13 : if Lin = True
14 :       Status ← "Requests over flooded"
15 :       end Program
16 : else if Cline[29] = 0
17 :       ActionFrame(Clean)
18 : else
19 :       ActionFrame(Malicious)
20 : return
```

System(): Standard function in C++ for executing system commands.
ActionFrame(): Displays the status of the file, i.e. Malicious or Safe.
TextStream(): Constructor of QTextStream class. It takes a parameter of the format QString, which is used to specify the destination of the input buffer, i.e. text file.
CurlPost() and CurlGet(): Standard procedure in Curl library, which is used to transfer and receive a file to and from a specified destination on the network.
Read(): Standard function in Qt of the header file <QFile.h>, to read a given text file using its path.

5 GRAPHIC USER INTERFACE OF THE PROGRAM



6 ADDITIONAL FEATURES IMPLEMENTED

6.1 Raw Details

The detected malicious file is already posted on the VirusTotal API, and the using the GET function of the API, the details about the malware are extracted and stored in a text file, while the internet scan function runs. The contents of this text file are displayed when the user clicks on the 'Raw Details' button.
The raw details about a malware include the type of malware, malware version and malware update according to multiple top antivirus software, including nProtect, QuickHeal, McAfee and TheHacker.

6.2 Examine Strings

For advanced users, we have added the provision to examine the strings in the file that is being checked, for the user to check exactly what the file is doing. This will be of minimal value to normal users, but advanced users will be able to check for malicious lines of code, even if the updated database does not contain the corresponding hash details.

6.3 Deletion Tab

This tab shows up only when a malicious file or directory is detected, with an option to delete the file from within the application.

6.4 Progress Bar

We have implemented a progress bar in our program, which makes it easier for the user to see the status of an operation being performed, and aids in gauging the amount of time remaining to complete the operation.
The progress bar also helps convey to the user that a process is running, and the program is working as intended.

7 ADVANTAGES OF THE PROJECT

Our project has the following advantages, which make it a useful application to have in one's computer:

1. Extremely quick: Our application is extremely fast in checking the status (safe/malicious) of the file, as it has been completely programmed in C++.

2. Two modes for hash-matching: As there are two modes for hash-checking, i.e. Local and Internet scan, our application can search if a file is a malware, even without internet access.

3. Updatable Database: The local database can be updated to include the latest malicious hashes available at https://virusshare.com/hashes, hence, the user can be sure that the database contains the latest known malicious hashes.

4. Access to file source code: This helps advanced users check the code for any abnormalities or unnecessary code.

5. Access to extensive online database: Our program allows users to access additional information about a type of malware a file is.

8 CONCLUSION

In this work, we created a portable Malware Scanner using the concept of Hash Matching. We utilized two different techniques for scanning malicious files in the system, one using a local, updatable database, while the other using the VirusTotal API. Both techniques proved to be advantageous in their own rights, and their use depends on the user's needs. Both methods were efficient, and our Malware Scanner met the requirements effectively.

9 REFERENCES

1. McAfee Labs Threat Report November 2020, https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-nov-2020.pdf

2. R Rivest, The MD5 Message-Digest Algorithm, 1992, IETF Network Working Group.

3. Vladimir Omar Calderon Yaksic, A Study of Hash Functions for Cryptography, https://www.giac.org/paper/gsec/3294/study-hash-functions-cryptography/105433

4. VirusShare, https://virusshare.com/

5. VirusTotal API, https://github.com/virustotal/c-vtapi

6. F Catak, A Yazi, A Benchmark API Call Dataset for Windows PE Malware Classification, https://www.researchgate.net/publication/332897624_A_Benchmark_API_Call_Dataset_for_Windows_PE_Malware_Classification

7. Libcurl – The multiprotocol file transfer library, https://curl.se/libcurl/

8. Qt Documentation – QcryptographicHash Class, https://doc.qt.io/qt-5/qcryptographichash.html