Signature-based Virus Detection for Windows Operating Systems Using Java and PostgreSQL

Eliyas Sala

I. Introduction

Did you know that Windows is the most used operating system? In fact, 76.12% of computer consumers use Windows operating system in 2021 [4]. Since it has existed for such a long time compared to others, hackers and modern terrorists have found loopholes to manipulate computer systems running this operating system. They have created different malwares, like viruses, that are often undetectable by antivirus softwares [2]. The data given above also motivates individuals who seek business opportunities, like myself, to develop a software to secure this system. Although different antivirus softwares exist, most of them are ineffective and lack the consumers' trust. This research briefly discusses how these programs work and their tested performances. Then I propose my software-based solution with different development tools.

.

II. Related work

Sukwong et al. examines the most used AV softwares and the algorithmic implementation behind them. He/she first provides the distinction between Signature-based detection vs. Behavior-based detection [5]. Both are ways to detect virus and every AV product doesn't necessarily use the same method to do that. In Signature-based detection, AV product scans a file and assigns a unique identification to that file. For example, it could use hashing algorithms like MD5 to assign value. Then it evaluates based on patterns of that hashing by comparing it to a remote database containing viral characteristics. If a match exists, then that file is indeed virus infected. Behavior-based detection, another common implementation, analyzes the behaviors of that specific file instead of accuracy matching. It is important to note that there are many more other implementations that different AV softwares use, but we will focus on these two.

Sukwong et al. continues to discuss and points out the ineffectiveness of AV softwares. Based on an experiment conducted to test the responsiveness of selected AV softwares, their activities were monitored when intentionally given infected files. The softwares in the experiment were the following: Avast, Kaspersky, McAfee, Norton, Symantec, and Trend Micro. Avast performed the best by detecting 62.15% of malware. However, they couldn't fully detect all of them. This clearly shows we shouldn't completely rely on our AV softwares to keep our windows computers secure.

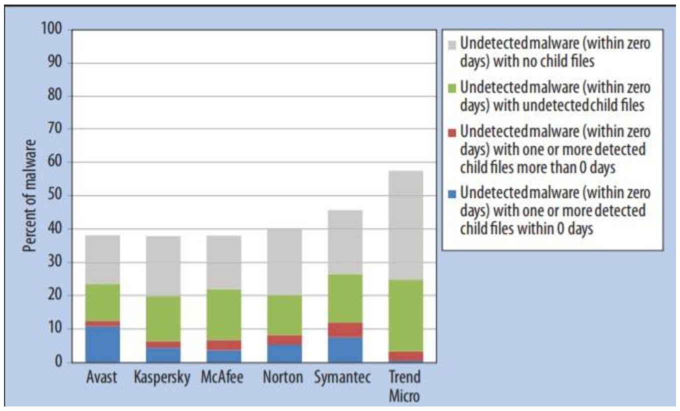


Figure 1 shows comparisons of different AV softwares

As you can see above, 42.83 percent of the malware remained undetected on first time. Among these undetected virus-infected files, 53.83 percent produced one or more malicious other files. Additionally, we found that 23.49 percent of these undetected malware files with malicious child files had one or more child files detected within zero days, but 64.36 percent weren't detected. This data tells us that there are limitations to these AV softwares.

III. Implementation

To solve the problem of enhancing virus detection capability on Windows operating systems, it is important to analyze the different types of viruses that exist and learn their behaviors. Based on the statistics provided on the global market share of desktop operating systems, Windows has consistently dominated the market for years in staggering amounts. Other well-known operating systems, like macOS, are far from reaching this success. However, this much of consumer usage has created vulnerability for Windows operating systems [4]. Unlike most antivirus software, my software application runs detection-only and does not have a removal feature. It builds upon a Signature-based detection system.

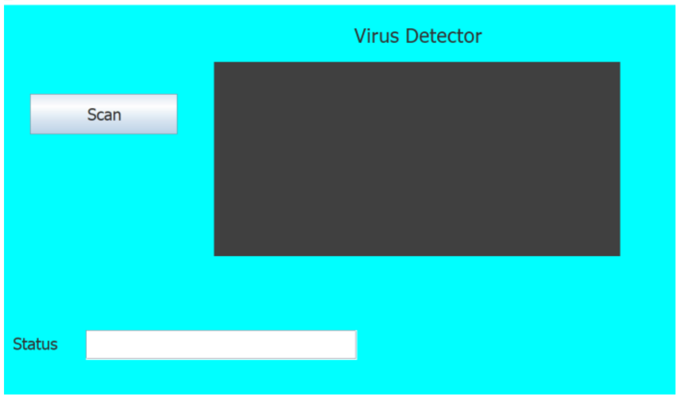


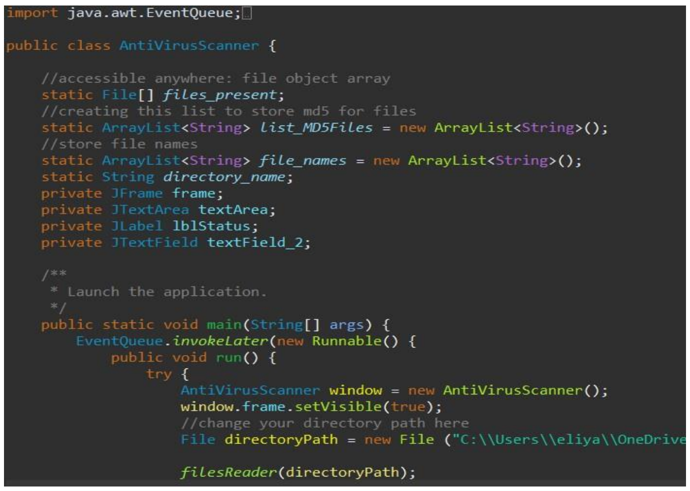
Figure 2 shows my software

Given a directory path to scan for viruses, my software will first use a hashing algorithm to compute the MD5 hashing for every file in that directory. Then it matches strings collections of stored MD5 data in a PostgreSQL database containing large data of the existing viruses and their MD5 classification signatures [3]. After cross-referencing the MD5 generated from scanning the files against our data of known viruses' MD5, my application decides whether is it safe or virus infected. If infected, a pop-up message should appear before the users warning them.

IV. WORK

Overall goal

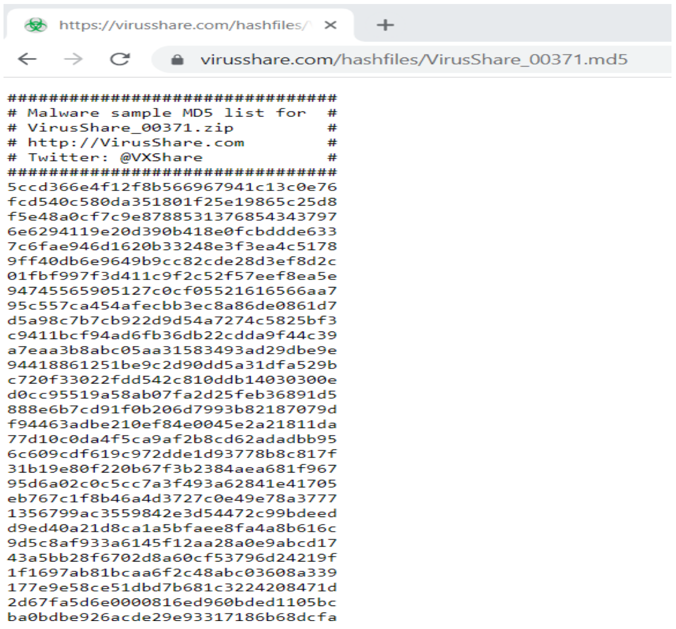
Deciding between Microsoft Visual Studio to develop a C/C# based program and Java-based desktop application. Since I have the most experience with Java, I chose to use Eclipse IDE to write my code and develop my program.



```java
import java.awt.EventQueue;

public class AntiVirusScanner {

    //accessible anywhere: file object array
    static File[] files_present;
    //creating this list to store md5 for files
    static ArrayList<String> list_MD5Files = new ArrayList<String>();
    //store file names
    static ArrayList<String> file_names = new ArrayList<String>();
    static String directory_name;
    private JFrame frame;
    private JTextArea textArea;
    private JLabel lblStatus;
    private JTextField textField_2;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    AntiVirusScanner window = new AntiVirusScanner();
                    window.frame.setVisible(true);
                    //change your directory path here
                    File directoryPath = new File ("C:\\Users\\eliya\\OneDrive

                    filesReader(directoryPath);
```

Figure 3 shows java libraries and data structures I used

I decided which programming langauge to use based on feasibility. I choose Java (my most proficient langauge) and I designed the user interface using Swing API. I created event listeners for the components as well. I imported Java's security class to use MessageDigest class to access it and generate MD5 hashing by scanning the files. The data structures I used include File object to store files as arrays and an arraylist to store MD5 strings. I also used a Scanner object to scan the files and store them as strings for manipulation.


https://virusshare.com/hashfiles/
virusshare.com/hashfiles/VirusShare_00371.md5

```
####################################
# Malware sample MD5 list for      #
# VirusShare_00371.zip             #
# http://VirusShare.com            #
# Twitter: @VXShare                #
####################################
5ccd366e4f12f8b566967941c13c0e76
fcd540c580da351801f25e19865c25d8
f5e48a0cf7c9e878853137685434379 7
6e6294119e20d390b418e0fcbddde633
7c6fae946d1620b33248e3f3ea4c5178
9ff40db6e9649b9cc82cde28d3ef8d2c
01fbf997f3d411c9f2c52f57eef8ea5e
94745565905127c0cf05521616566aa7
95c557ca454afecbb3ec8a86de0861d7
d5a98c7b7cb922d9d54a7274c5825bf3
c9411bcf94ad6fb36db22cdda9f44c39
a7eaa3b8abc05aa31583493ad29dbe9e
94418861251be9c2d90dd5a31dfa529b
c720f33022fdd542c810ddb14030300e
d0cc95519a58ab07fa2d25feb36891d5
888e6b7cd91f0b206d7993b82187079d
f94463adbe210ef84e0045e2a21811da
77d10c0da4f5ca9af2b8cd62adadbb95
6c609cdf619c972dde1d93778b8c817f
31b19e80f220b67f3b2384aea681f967
95d6a02c0c5cc7a3f493a62841e41705
eb767c1f8b46a4d3727c0e49e78a3777
1356799ac3559842e3d54472c99bdeed
d9ed40a21d8ca1a5bfaee8fa4a8b616c
9d5c8af933a6145f12aa28a0e9abcd17
43a5bb28f6702d8a60cf53796d24219f
1f1697ab81bcaa6f2c48abc03608a339
177e9e58ce51dbd7b681c3224208471d
2d67fa5d6e0000816ed960bded1105bc
ba0bdbe926acde29e93317186b68dcfa
```

Figure 4 shows the data source I used for identifying viruses

I used a real-world dataset from VirusShare.com. First, I stored the MD5 data in text files and then, I used Heroku to create a PostgreSQL database and added database driver to eclipse. Then I wrote a class for database connection and data insertion. Every time my software runs, it should either return "clean" or "virus detected". When users click "scan" button, it runs the scanning process to scan for viruses. The backbone of my application is the MD5 hashing function and Checksum functions that I wrote. I created a byte array to read data in chunks. Note that "checksum" function as a parameter of previous function's data. I incorporated multiple try and catch blocks since it is a good design pattern and conventional way to write a good code.

V. CONCLUSION

This research-based project explores the different virus detection systems for Windows operating system. Then it discusses a signature-based scanning system with its implementation. Although my database only contains 10,000 rows of data due to resource limitations, the application is fully scalable and will continue to undergo refinement until my desired goals are met and optimal solutions are achieved.

References

[1] Al-Asli, Mohammed et al. “Review of Signature-based Techniques in Antivirus Products.” (2019).

[2] Garba, Faisal A. et al. “Evaluating the State of the Art Antivirus Evasion Tools on Windows and Android Platform.” (2019).

[3] Sahoo, Abhaya, et al. “Signature based Malware Detection for Unstructured Data in Hadoop.” (2014).

[4] “Desktop operating system market share worldwide,” StatCounter Global Stats. [Online]. Available: https://gs.statcounter.com/os-market-share/desktop/worldwide. [Accessed: 05-Sep-2021].

[5] O. Sukwong and H. S. Kim, "Commercial Antivirus Software Effectiveness: An Empirical Study," IEEE Computer Society, pp. 63-70, 2011.