Technique to thwart the opening of a virus embedded file without the aid of an anti-virus software.

Sriram Kalyanaraman

Sri Venkateswara College of Engineering, Sriperumbudur, Chennai, India

Summary
A technique devised to prevent the spread of virus by first detecting it and then stopping it's execution by not allowing the infected file to open in the first place. The technique is capable of carrying out this functionality without using an anti-virus program.

Key words:
virusStopper ,Virus signature

1. Introduction

A **virus** by definition is a computer program that can replicate itself and spread from one computer to another or possibly even within the same computer by the opening of certain previously infected files. Most viruses are embedded in other type of files probably image, text or video files. When these are executed (i.e. opened) the replication process starts. But the most vital point to note is that, viruses remain harmless as long as they are left alone by not allowing them to execute. But this is not always possible because novice users or sometimes even expert users tend to open these viruses embedded files with or without intention. So, this makes the anti-virus software an absolute necessity. But this newly proposed technique provides the much needed protection without anti-virus software.

2. Tables, Figures and Equations

2.1 Tables and Figures

Table 1:Binary values of a sample virus

Sample virus code	1	0	1	0	0	0	1	1	1	0

Table 2:Randomly generated binary values

Randomly generated sequence	1	1	0	1	1	0	0	1	1	1

Table 3:Sequence of 0's is generated

Composed entirely of 0's	0	0	0	0	0	0	0	0	0	0

Table 4:Sequence of 1's is generated

Composed entirely of 0's	1	1	1	1	1	1	1	1	1	1

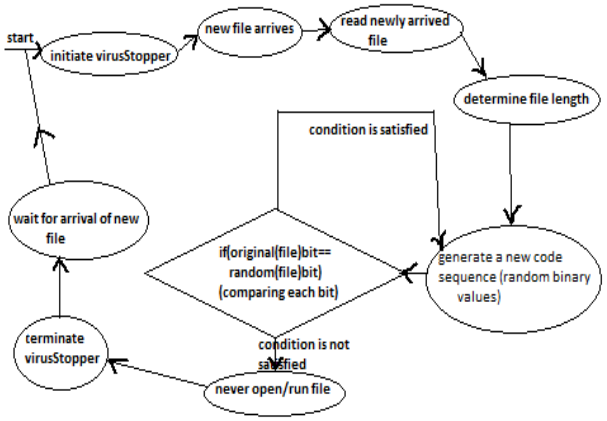

Fig. 1 Working of the virusStopper

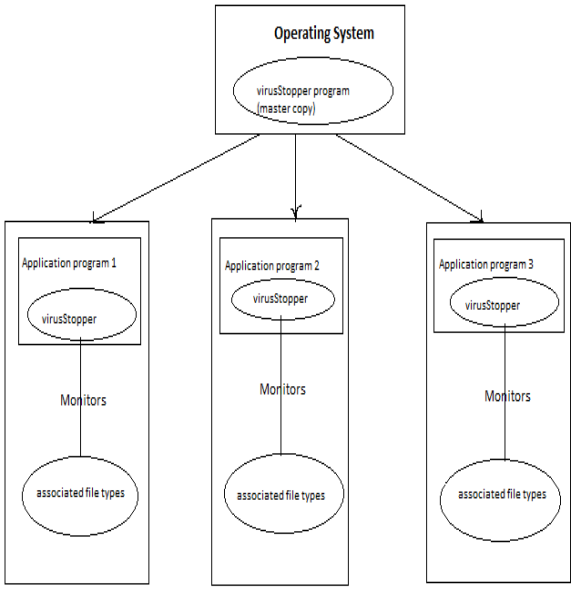

Fig. 2 Proposed Architecture of the replication process of virusStopper

Manuscript received September 5, 2011
Manuscript revised September 20, 2011

3. Paragraphs and Itemizations

A virus code like all other programs is composed entirely of binary values. The **virusStopper** code contains the following modules:

Module 1:
It reads a file and searches for the virus pattern in it. Virus patterns also known as virus signatures are stored in separate files (something like a database). If a pattern is found in the original file that matches any of the virus signatures, Module 2 is called.

Module 2:
Catches the virus embedded file and calculates the length of the file (or the length of the virus pattern).

Module 3:
This module calls the BitGenerator() method. This method generates a sequence of random bits whose length is equal to that of the original file's length. Refer to Table 2 for the sample values of the randomly generated code.

Module 4:
This contains equals () method. It compares the original file and the randomly generated file binary bit by bit. If at least one bit differs then the file is never opened.
At the worst case if all the bits match, then Module 5 is called.

Module 5:
When both the original and the randomly generated files match at every bit, virusStopper generates a new sequence based on the resources available,

Case1:
A set of new random binary values (Refer to Table 2*)* to compare with the original file is generated and the comparison of the two files is done again. This process is repeated until a match never occurs. This process is iterative in nature.

Case2:
A sequence of complete 1's is generated (Refer to Table 3) and the comparison is made. This method will be effective because no code (virus code) will be composed of the same bit entirely.

Case 3:
A sequence of complete 0's is generated (Refer to Table 4) and the comparison is made. Again, this method will be effective because no code (virus code) will be composed of the same bit entirely.

A sample code is written in Java to implement the above said concept of thwarting the opening of a virus embedded file.

```
// Programmatic representation of Fig. 1
import java.io.*;
public class ReadStringFromFile {
public static void main(String[] args) {
File file = new File("C://sample//samplevirus.txt");

int ch;
StringBuffer strContent = new StringBuffer("");
FileInputStream fin = null;
try{
fin = new FileInputStream(file);
while( (ch = fin.read()) != -1)
strContent.append((char)ch);
fin.close();
}
catch(FileNotFoundException e{
System.out.println("File " + file.getAbsolutePath() +
                " could not be found on filesystem");
}

catch(IOException ioe){
System.out.println("Exception while reading the file" + ioe);
}
System.out.println("File contents :");
System.out.println(strContent);
System.out.println(strContent.length());
BitGenerator bg = new BitGenerator();
String generateBits = bg.generate(strContent.length());
System.out.println(generateBits);

while(strContent.equals(generateBits)) {
generateBits = bg.generate(strContent.length());
}
}
}
```

Explanation for the above code is as follows,
A file is created and binary values are entered. This is assumed to be the sample virus code under study. The file is opened and the contents are read character by character.
During this process, the file is searched for any of the known virus signatures .These signatures may be stored on a database or a file system. When the end of file is reached, the program calculates the length of the sample input file.

```
//Random Bit Generator
import java.util.Random;
public class BitGenerator{
public static void main(String[] args) {
String bits = "";
Random r = new Random();
```

```
for(int i=0; i<10; i++){
int x = 0;
if(r.nextBoolean()) x=1;
bits += x;
}
System.out.println(bits);
}
}
```

And the functionality of Random Bit Generator program is to generate a random sequence of binary values equal to the length of input file. A new random sequence will be generated until the contents of the sample input file and the randomly generated file do not match. The above demonstrated programs are bundled together and is referred as the virusStopper.

The best place for the implementation of virusStopper would be within the operating system .Whenever a new application program is started, virusStopper places a copy of itself within that application program (Refer to Figure 2) .It then scans all the file extensions associated with that application program. When many programs are opened simultaneously, virusStopper replicates itself from the operating system. The basic difference between a virus and virusStopper is mentioned below,

Virus: Replicates itself to cause damage to the system.
virusStopper: Replicates itself to prevent damage to the system.

So a distributed environment is created which makes the whole virus thwarting process a highly efficient one.

The virusStopper program is present in the main memory. But once the application program is erased from the main memory, the virusStopper code is also erased. So, the resources held by it previously will be automatically released. The biggest advantage of this technique is it's simplicity and that the installation of virusStopper into the operating system is a onetime process and works efficiently as long the operating system is used.

References

[1] http://en.wikipedia.org/wiki/Computer_virus
[2] www.stackoverflow.com
[3] http://www.pamukcular.com/?p=102

Sriram Kalyanaraman is a Final year student of B.Tech Information Technology at Sri Venkateswara College of Engineering, Sriperumbudur, Chennai, India.