IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Study of CFG and System calls for Computer Virus Detection

Ankur Singh Bist
Quantum Global Campus, Roorkee, India
ankur1990bist@gmail.com

Abstract

Computer viruses are big threat to computer world; researchers doing work in this area have made various efforts in the direction of classification and detection methods of these viruses. Graph mining, system call arrangement and CFG analysis are some latest research activities in this field. The computability theory and the semi computable functions are quite important in our context of analyzing malicious activities. A mathematical model like random access stored program machine with the association of attached background is used by Ferenc Leitold while explaining modeling of viruses in his paper. Computer viruses like polymorphic viruses and metamorphic viruses use more efficient techniques for their evolution so it is required to use strong models for understanding their evolution and then apply detection followed by the process of removal. Code Emulation is one of the strongest ways to analyze computer viruses but the anti-emulation activities made by virus designers are also active. This paper involves the study of control flow graphs and system calls used for detection of computer viruses in better manner.

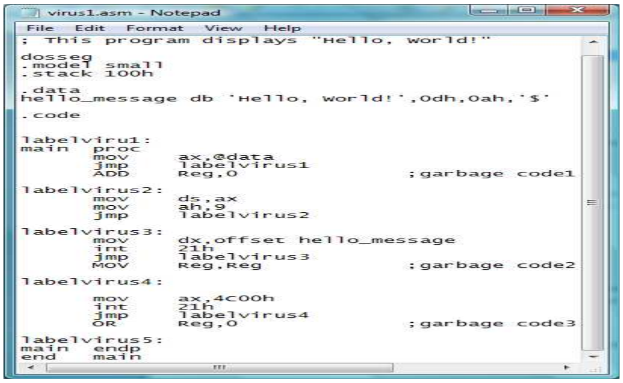
Keywords: Control Flow Graph, Malicious Codes.

Introduction

There are various processes that have been used in the direction of classification of computer viruses from normal files that will finally lead to worm detection. Machine learning techniques are widely used in this direction. As statistics says that the attacks of malicious codes are increasing day by day so there is requirement of strong techniques that can be used for their detection. Malicious code designers use lot of techniques that are difficult to analyse and detect. The static methods also seems not to work in the case where every time there are rapid dynamicity from attacker side so now a days main focus is going on towards the methods that are dynamic and are able to detect zero day computer viruses.

The rise in the malicious threats like computer viruses activities are required to be handled and observed strongly to make certain defence that can stand as a saviour of security domain. Other types of malware are:

1. Worms
2. Trojan horse
3. Botnets
4. Adware
5. Spyware



```
; This program displays "Hello, world!"

dosseg
.model small
.stack 100h

.data
hello_message db 'Hello, world!',0dh,0ah,'$'

.code

labelviru1:
main   proc
       mov     ax,@data
       jmp     labelvirus1
       ADD     Reg,0                   ;garbage code1

labelvirus2:
       mov     ds,ax
       mov     ah,9
       jmp     labelvirus2

labelvirus3:
       mov     dx,offset hello_message
       int     21h
       jmp     labelvirus3
       MOV     Reg,Reg                 ;garbage code2
labelvirus4:

       mov     ax,4C00h
       int     21h
       jmp     labelvirus4
       OR      Reg,0                   ;garbage code3

labelvirus5:
main   endp
end    main
```

Figure1. Assembly file of virus

The mutating behaviour of metamorphic viruses is due to their adoption of code obfuscation techniques.

a) Dead code insertion
b) Variable Renaming
c) Break and join transformation
d) Expression reshaping
e) Statement reordering

System Call and Control Flow Graph

In computer science the process by which program requests a service from an operating system kernel is called system call. This may include hardware related services like accessing the hard disk, creating and executing new processes, and communicating with integral kernel services e.g. scheduling. An important interface between a process and the operating system is introduced by system calls. Implementing system calls requires a control transfer which involves some specific kind of features from architecture. A complex method to implement this is to utilize a software interrupt or trap. Interrupts gives control to the operating system kernel so software simply require to set up some register with the system call number needed, and make the execution of software interrupt.

This is the only technique provided for many RISC processors, but CISC architectures such asx86 support some other methods. One example is SYSCALL/SYSRET, SYSENTER/SYSEXIT. The two mechanisms were independently designed by AMD and Intel, respectively, but in essence do the same thing. These are "fast" control transfer instructions that are designed to quickly transfer control to the OS for a system call without the overhead of an interrupt. Linux 2.5 began using this on the x86, where available; formerly it used the INT instruction, where the system call number was placed in the EAX register before interrupt 0x80 was executed.

An older x86 mechanism is called a call gate and is a way for a program to literally call a kernel function directly using a safe control transfer mechanism the OS sets up in advance. This approach has been unpopular, presumably due to the requirement of a far call which uses x86 memory segmentation and the resulting lack of portability it cause, and existence of the faster instructions.

For IA-64 architecture, EPC (Enter Privileged Mode) instruction is used. The first eight system call arguments are passed in registers, and the rest are passed on the stack.

In the IBM System/360 mainframe family, a Supervisor Call instruction implements a system call for legacy facilities; the Program Call instruction is used for newer facilities. In particular, PC is used when the caller might be in SRB mode.

There are five major categories of system call:

1. Procedure of process control
   - loading
   - Execution
   - Create process (for example, fork on Unix-like systems or Nt-Create Process in the Windows NT Native API)
   - Process termination
   - Get/Set process attributes
   - Wait for time, wait event, signal event
   - Allocate, free memory
2. Procedure of file management
   - Create file, delete file
   - Open, close
   - Read, write, reposition
   - Get/set file attributes
3. Procedure of device management
   - Request device, release device
   - Read, write, reposition
   - Get/set device attributes
   - Logically attach or detach devices
4. Procedure of information maintenance
   - Get/set time or date
   - Get/set system data
   - Get/set process, file, or device attributes
5. Procedure of communication
   - Create, delete communication connection
   - Send, receive messages
   - Transfer status information
   - Attach or detach remote devices

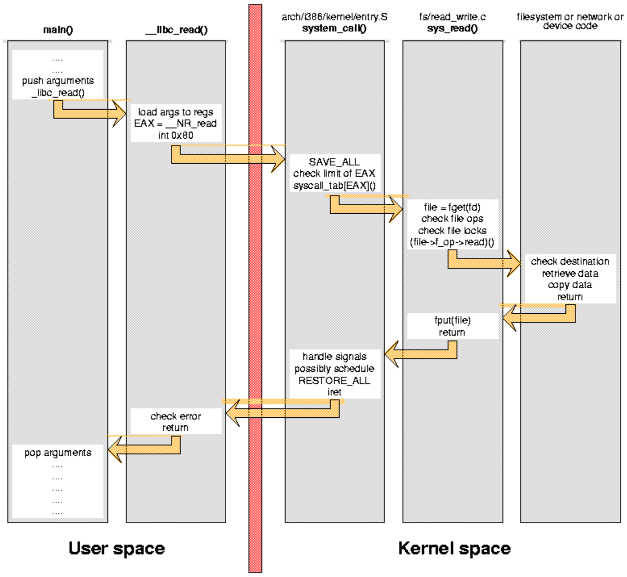


Figure 2:- system call procedure

In computer science control flow graph is defined as a presentation, using graph notation, of all paths that might be traversed through a program during its execution.

In a control flow graph each node in the graph represents a basic block, i.e. a straight-line piece of code without any jumps or jump targets; jump targets start a block, and jumps end a block. Directed edges give information of jumps in the control flow. There are, in most presentations, two specially designated blocks:

Entry block: - By which control enters into the flow graph.

Exit block: - By which all control flow leaves.

Control flow graph is widely used in compiler optimizations and static analysis tools. The research involves system call and CFG analysis considers the system call and CFG pattern of normal files and malicious files. The main purpose of researchers remains in finding the difference their system call or CFG pattern. The analyzed difference becomes an important measure for classification. In this way the computer virus detection problem reduces into mathematical problem of finding similarity in specific terms like isomorphism in graphs.

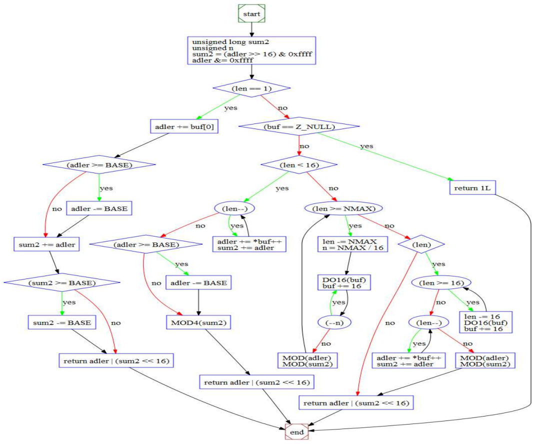


Figure 3: Control flow graph

Conclusion

This paper discusses about basic outline of computer viruses and their detection by analyzing system call and control flow graphs. The methods discussed are being used for solving different problems in this domain. This study will be helpful for researchers working in the field of computer virology.

References

[1] www.wikipedia.com.

[2] Christian Wressnegger,"Beatrix: A Malicious CodeAnalysis Framework".

[3] S. Papadimtrou and J. Sun. Disco: distributed co clustering with map reduce in proceedings of ICDM, 2008.

[4] Farrokh Mamaghani ,Evaluation and selection of an antivirus and content filtering software Department of Management, St John Fisher College, Rochester, New York, USA)