CHAPTER 3

FRAMEWORK FOR IDENTIFICATION OF NON-SELF OPERATING SYSTEM PROCESS

Computers have proved to be an unavoidable part of our modern life. Every work of contemporary life involves directly or indirectly the use of computers. A lot of personal as well as private work related information is stored in computer systems. Therefore, it is an imperative undertaking to protect this information and protect it. In this chapter, the aim is to ascertain the sense in a computer system that could differentiate between the Self processes (i.e. Processes that are not harmful to our computer system) and the Non-Self processes (i.e. Processes that are harmful and treacherous to computer system). A process coming into the system is identified whether the process is part of the stable system, i.e. Self process or is it a harmful process which can wallop at the basics of a system i.e. Non-Self process. This is through with the help of the Detectors (Decision Tree/Artificial Neural Network) generated by the genetic algorithm. These techniques would be used to categorize the operating system processes into the Self (Non-harmful) and Non-Self (harmful or dangerous). This would help the system to sense the processes before the harmful processes do any harm to the system.

3.1 Introduction

The resemblance between Computer Security problems and biological processes was recognized as early as 1987. Later, Eugene H. (1994) argued that computer viruses are a form of artificial life, and several authors investigated the equivalence between epidemiology and the burden of computer viruses across networks. However,
current methods for protecting computers against viruses and numerous other kinds of intrusions have mostly failed to take benefit of what is known about how natural biological systems protect themselves from infection. Some prelude work in this inclination integrated a virus detection method based on T-cell censoring in the thymus and an incorporated approach to virus detection incorporating ideas from various biological systems. However, these early efforts are normally regarded as novelties, and the ideology they demonstrate have yet to be commonly adopted.

Immunologists conventionally described the dilemma solved by the immune system as that of distinguishing “Self” from dangerous “other” (or “Non-Self”) and eliminating other. Self is taken to be the internal cells and molecules of the body, and Non-Self is any unfamiliar substance, mainly bacteria, parasites, and viruses. The difficulty of defending computer systems from malevolent intrusions [Dasgupta D. (1999)] can similarly be viewed as the problem of distinguishing Self from dangerous Non-Self [Lyanne W. (2002)]. In this case, Non-Self nerve is an illegal user, unfamiliar code in the form of a computer virus or worm, unforeseen code in the form of a Trojan horse, or besmirched data.

A novel computational intelligence technique, inspired by immunology [Farmer J. et al. (1986)], has emerged, called Artificial Immune Systems [Steven A. and Forrest S. (2000), Sabine B. (2006)]. Artificial Immune Systems (AIS) are relatively young promising techniques, which discover, originate and pertain diverse biologically inspired immune mechanisms, aimed at computational problem solving [Farmer J. et al. (1986), Carter J. (2000)]. Several concepts from the immune have been extracted and applied in solution to real world science and engineering problems [Lyanne W. (2002)].

AIS also used to provide Security to a computer system. The approach is to design a tool for computer system that could distinguish a Non-Self process from the Self process. Self process means part of the stable system and which do not harm the system [Percus J. et al. (1993), Forrest S. et al. (1994 ,1996)]. These processes are generally user or system processes executing on a computer system. Non-Self processes are the processes that use CPU resources inappropriately and are harmful to the system. These processes have properties of making themselves a part of the computer system, hide their original identities and replicate themselves into a large number to extensively use system resources [Percus J. et al. (1992), Jinzhong N. et al. (2003)]. Non-Self means which makes the system unstable and harm it.
The aim of this chapter is to describe a methodology to identify the Self and Non-Self operating system’s process. The concepts of Machine Learning are used to describe the proposed framework. The concepts like Decision Tree (DT), Artificial Neural Network (ANN) and Genetic Algorithms (GAs) are used within the framework to identify the Non-Self processes. Decision Tree Learning the learned target function is precise by Decision Tree which provides decision through root to a leaf node. Artificial Neural Network provides a proposal for learning vector-valued, real-valued and discrete valued functions from a given set of examples.

GAs based on the evolutionary ideas of natural selection and genetics. GA is always gives the optimized results [Eiben A. et al. (1994), Rojas R. (1996)]. The relevance of the work done in this chapter comes in computer security. The problem of Computer Security can be seen as the problem to distinguish between Self and Non-Self. The main task is to find Non-Self processes, to give notification to the user and pause or terminate the process. The main aim is to design such system for computer security that tries to draw an analogy between computers and biological systems. The biological equivalent to the similar task is the immune system. The proposed system is modeled on the human body and is capable of recognizing and monitoring its own state of Self.

### 3.2 Proposed Methodology

The concept of Artificial Immune Systems is used in this chapter with the inspiration of the Natural Immune System [Steven A. and Forrest S. (2000), Divyata D. et al. (2008)]. The area of Artificial Immune Systems (AIS) deals with the study and development of computationally interesting abstractions of the immune system. Artificial Immune Systems are widely used for solving problems from mathematics, science, engineering, and information technology. Artificial Immune System is a part of biologically inspired computing and natural computation with interests in machine learning and it belongs to the broad area of Artificial Intelligence [Dasgupta D. (1999), Saber D. et al. (2006), Divyata D. et al. (2008)]. In this chapter, an approach is proposed based on the Machine Learning. A modular design is proposed to have four modules [Kumar A. and Kumar S. (2014a)]. A fully developed system will be able to identify or isolate Self and Non-Self processes of a computer system. The operating system processes would be checked by the proposed framework before it enters in the ready state (executed by the CPU).
**Structure of Proposed Algorithm**

The complete approach of process identification can be divided into four main modules as shown in Figure 3.1. These four modules are:

- Generation of Detectors with the help of genetic algorithm / Construction of Decision Tree / Artificial Neural Network
- Training of the Detectors for identification of Self and Non-Self process / Learning of Decision Tree / Artificial Neural Network
- Identification of Self and Non-Self process in actual environment.
- Action taken against the Non-Self process

---

**Figure 3.1:** Structure of the proposed framework

The output of the first module is the Detectors / Decision Tree / Artificial Neural Network. These Detectors / Decision Tree / Artificial Neural Network are the input of the second module. In second module training is provided to these Detectors / Decision Tree / Artificial Neural Network. The output of second module is the trained Detectors / Decision Tree / Artificial Neural Network. Now these trained Detectors / Decision Tree / Artificial Neural Network become the input of the third module. In third module the trained Detectors / Decision Tree / Artificial Neural Network work in actual environment. Third module provides
a list of Self and Non-Self operating system processes as output of this module. The output of third module will become the input of the last module. In fourth module the user takes action on Non-Self processes. User can suspend, terminate or kill the Non-Self processes. User can resume the Non-Self processes and define this Non-Self process as Self-process. (In Appendix-I users actions on Non-Self processes are discussed.)

3.2.1 Module 1: Generation of Detectors / Construction of DT /ANN

This module deals with generation of Detectors / Construction of Decision Tree /Artificial Neural Network which will be used to detect any instance of abnormal behavior of the system. For construction of Decision Tree the concept of Entropy, Information Gain and ID3 algorithm is used. For the construction of an Artificial Neural Network, input, hidden and output nodes are defined. The generation of Detectors will be done using a genetic algorithm. This module will have two sub modules:

- **Detector Generator**
  
  Detectors are generated with the help of genetic algorithm. A Genetic algorithm will generate Detectors randomly. The Detectors are not present in the data collected in the process, but they are very close to the original sequences.

- **Detector Verification**
  
  Each Detector will be first matched with all the recorded sequences. The Detector will only be accepted if it does not match any of the recorded sequences, if it matches any of the sequences then it will be rejected. This is done to protect against auto-immunity, i.e. the Detector should not match with a sequence of normal behavior.

Figure 3.2 shows the working of the first module. In this figure first we analyze the process and its structure. The main focus is on the process parameters. Parameters which are not null and have useful information about processes like Process Id, File Size, Base Address, Memory Usage, Number of Page Faults, Page File Usage, etc. Then the first set of Detectors is randomly generated as according to the selected process parameters and their structure. These randomly generated Detectors then match with the process structure for checking the range and size of the parameters. If there is a match the randomly generated Detectors are selected for the next module. If there are missing-match, then these randomly Detectors are discarded. To generate the new set of Detectors the genetic crossover operators are used.
Algorithm of module 1

- Step 1: Analyze the process structure (according to the parameter selected) of a process for important and relevant parameters.
- Step 2: Generate the Detectors randomly for creating the first Detector set according to the process structure / construct a Decision Tree / Artificial Neural Network.
- Step 3: The Detectors generated in Step 2 are matched to the process structure.
  - Step 3.1: If the Detector does not match with the process structure then it is discarded.
  - Step 3.2: If the Detector matches with the process structure, then it is added to the Detector database.
- Step 4: If the Detector is added to the Detector database, the new Detectors are cloned (by using a Genetic Crossover operator) to the existing Detector database.
- Step 5: Go to Step 3 until sufficient Detector database is generated.

Figure 3.2: Flow Diagram for Module 1

3.2.2 Module 2: Training to Detectors set / DT / ANN for Identification of Self and Non-Self process

In this module, evaluate the Detector set / DT / ANN and necessary training is provided so that it covers all details as according to the process structure. A complete trained Detector set
/ DT ANN would be the outcome at the end of this module. Figure 3.3 shows the working of module 2. In this module fitness of every Detectors generated in module 1 is checked. It may happen that some Detectors have same parameter values, so these Detectors are discarded. After checking the fitness of the Detector, training is provided to Detectors by using a training data set.

**Algorithm of module 2**

- **Step 1:** Take each of the candidate Detectors one by one and check whether the Detector is covering all the details as according to the process structure.
  - Step 1.1: If the candidate Detector does not cover all the details, it is removed from the Detector database.
  - Step 1.2: If the candidate Detector covers all the details, it is evaluated for its quality.

- **Step 2:** For evaluating the quality of the Detector, it is checked that if it is overlapping with another Detector or not.
  - Step 2.1: If the Detector overlaps, it is removed from the Detector database.
  - Step 2.2: If it doesn’t overlap, go to next Step No.3

- **Step 3:** Take Sample Self processes and match the process structure of these sample Self processes with the candidate Non-overlapping Detector.
  - Step 3.1: If it matches, the candidate Detector is discarded.
  - Step 3.2: If it doesn’t match, this candidate Detector is added to the trained Detector set.

**3.2.3 Module 3: Identification of Self and Non-Self Process in Actual Environment**

In this module, the system would be performed against a process in the actual environment. If the process is detected by the Detector, it would be treated as a Nob-Self process and the User would be intimated about the result.

Figure 3.4 shows the working of the module 3. The trained Detectors from module 2 are used to identify the Non-Self processes in actual environment. If any process is identified as Non-Self, we check that the child, the Non-Self processes first.
Trained Detectors/ Learned System

Take process from the Real environment

For Each candidate Detector

Is covering all details as according to process structure

Evaluate quality of Detector

Is Overlapping Detector

Yes

No

Remove the Detector from the database

Yes

No

Add to Trained Detectors set

Discard the Detector

If Matches

Yes

No

Take sample self process and match the structure of the process with the Detector

Figure 3.3: Flow Diagram for Module 2

Trained Detectors/ Learned System

Take process from the Real environment

Is Identified by the Detector

Yes

No

Is the Non-Self Process create its child process

Add the process in Non-Self process list and inform the user to take action

Go for another process

Figure 3.4: Flow Diagram Module 3
Algorithm of module 3

- Step 1: Take the trained Detectors / Decision Tree / Artificial Neural Network.
- Step 2: Take a process from the actual current environment.
- Step 3: Check whether the process is identified by any of the trained Detectors.
  - Step 3.1: If the process is not identified by the trained Detectors then leave that process and go to Step 5.
  - Step 3.2: If the process is identified by the trained Detectors/ Decision Tree /Artificial Neural Network then this process is added to a list of Non-Self processes and the user is informed to take an action.
- Step 4: The processss in the Non-Self list is checked whether it has created child processes or not.
  - Step 4.1: If yes, then take that child process and proceed from Step No. 3
  - Step 4.2: If not, then go to Step No.2
- Step 5: Go to Step 2 until all the processes have been checked once.

3.2.4 Module 4: Action Taken Against the Non-Self Processes

In this last module, an action environment is provided to the user where he can select the action to be taken against the Non-Self processes. The actions that a user can take includes are deletion of the Non-Self Process and / or suspension of working of the Non-Self Process. In appendix-I users actions on Non-Self processes are discussed. Figure 3.5 shows the user’s action on Non-Self processes. User can delete or suspend the process. (In Appendix-I users actions on Non-Self processes are discussed.)

Algorithm of module 4

- Step 1: Take one process from the list of Non-Self processes.
- Step 2: Ask the user whether he wants to delete this process or not.
  - Step 2.1: If the user wants to delete the Non-Self process, then delete the process and go to Step No.3
– Step 2.2: If the user does not want to delete the Non-Self process under consideration, then the process is left in the Non-Self processes list with its working being suspended.

• Step 3: Go for another process from the list of Non-Self processes.
• Step 4: Go to Step 2.
• Step 5: Repeat until the user is asked for his choice for all the Non-Self processes.

---

3.3 Comparison with existing approaches

The proposed framework for identification of Non-Self operating system processes works on the lowest level (i.e. on process parameter level). If any computer system will be infected by the attacks or viruses, then its process will be generated by the operating system. The main

---

Figure 3.5: Flow Diagram Module 4
Figure 3.6: Comparison of Scan Time of Few Antivirus Tools with Proposed Framework

Figure 3.7: Comparison of Accuracy of Few Antivirus Tools with Proposed Framework
Figure 3.8: Comparison of Detection Rate of Few Antivirus Tools with Proposed Framework

Figure 3.9: Comparison of Performance Lag due to Few Antivirus Tools with Proposed Framework
Figure 3.10: Comparison of Security by Antivirus Tools with Proposed Framework

Table 3.1: Outcome of Proposed Framework in comparison to few Antivirus tools

<table>
<thead>
<tr>
<th>Antivirus</th>
<th>Parameters</th>
<th>Scan Time</th>
<th>Accuracy</th>
<th>Detection Rate</th>
<th>Performance Lag</th>
<th>Signature based Detection</th>
<th>Regular Updating Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG antivirus</td>
<td></td>
<td>High</td>
<td>High</td>
<td>Normal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Norton antivirus</td>
<td></td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Avast antivirus</td>
<td></td>
<td>High</td>
<td>Medium</td>
<td>Normal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MS Security Essentials</td>
<td></td>
<td>Average</td>
<td>High</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Proposed Approach</td>
<td></td>
<td>Low</td>
<td>Very High</td>
<td>Very High</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
aim of the proposed framework is to identify these processes. Various concepts of Soft Computing and Machine Learning can be implemented in this framework. On the other hand the existing approaches work on the stored data by comparing this stored data with the existing signature database. A regular updating of the antivirus tool is required, but in the proposed framework, there is no need of it. To provide the Security at the lowest level (process level) the approach proposed in the chapter will be more effective than the other approaches as it works on processes parameters to identify the Non-Self (harmful) processes.

Figure 3.6 shows the comparison of scan time of few antivirus tools with proposed framework. For Figure 3.6 the test conducted on default settings, time to scan 4500 of data for antivirus tool and time to scan all processes. Figure 3.7 shows the comparison of accuracy of few antivirus tools with proposed framework.

Figure 3.8 shows the comparison of detection rate of few antivirus tools with proposed framework. Figure 3.9 shows the comparison of performance lag due tofewerw antivirus tools with proposed framework. Table 3.1 shows summarized comparison result in Figure 3.6, 3.7, 3.8 and 3.9. Table 3.1 also shows that the performance lag, signature based detection and regular updating required of the few anti – virus tool available in the market with the proposed approach framework.

After analyzing it is found that the proposed framework is better than the others. Figure 3.9 shows the comparison of security (% ration of infected file identified) achieved by some popular antivirus tools with the proposed framework. It is clear from the Figure 3.10 that the proposed framework provides better security as proposed approach scan only the operating system processes.

3.4 Summary

This chapter deals with the identification of Self and Non-Self operating system process to provide the highest level of Security for a computer system. The proposed technology can be applied to any type of computer system to provide the highest level of Security. Since the work is done at the process level so a higher level of Security will be provided by the proposed framework. The proposed framework can be used by the concepts of Machine Learning and Soft Computing techniques. In this chapter, the work is divided into four modules, which are a generation of Detectors / construction of Decision Tree/ Artificial
Neural Network, training of the Detectors Decision Tree/ Artificial Neural Network, identification of the Self and Non-Self processes, and the notification to the user.

In the first module, a set of Detectors is randomly generated according to the process structure of a Self process (Decision Tree/ Artificial Neural Network are constructed). Then, in the second module, this Detector set / Decision Tree/ Artificial Neural Network are trained to identify the Non-Self processes in the real environment and training the Detector to identify the Non-Self processes.

In the next module, this trained Detectors set Decision Tree/ Artificial Neural Network are used in the real environment to identify the Non-Self processes and in the last module, the user is intimidated about the Non-Self processes and is given a choice of whether he wants to delete the process or suspend the working of the Non-Self process. (In Appendix-I users actions on Non-Self operating system processes are discussed in details.)

Since there are not many applications available on this concept, our system is relatively new. It is capable of identification of Self and Non-Self processes and it notifies the user. It monitors the specified process of the system for changes which might make the process harmful or due to which the process might be affecting the stability of the system. It does away with the idea of using virus databases for identification of viruses. The plan is to design a tool based on the concepts of Machine Learning to identify Self and Non-Self process and its subparts. After the full design of the tool, the performance of the system will be analyzed.

Decision Tree is successfully implemented within the proposed framework. Chapter 4 of the thesis explains how the concepts of Decision Tree are used within the framework to provide the Security to the computer system by filtering the Non-Self process. Artificial Neural Network is successfully implemented within the proposed framework. Chapter 5 of the thesis explains how the concepts of Artificial Neural Network are used within the framework to provide the Security to the computer system by identifying Non-Self process.

Genetic Algorithm is successfully implemented within the proposed framework. Chapter 6 of the thesis explains how the concepts of Genetic Algorithm and its operators are used within the framework to provide the Security to the computer system by identifying Non-Self process.
In appendix-I, the detail description of the module 4 “Action Taken Against the Non-Self Processes” is explained. An action environment is provided to the user where he or she can select the action to be taken against the Non-Self processes according to the operating system used by the system.