Article can be accessed online at http://www.publishingindia.com

Artificial Neural Network Based Approach for Identification of Operating System Processes

Amit Kumar*, Shishir Kumar**

Abstract

A computer system can be secured by using various methods like firewalls, anti-virus tools, network security tools, malware removal tools, monitoring tools etc. These tools and applications are being by most of the computer users. These computer security tools need to be updated and monitored regularly by the user. If any computer users fail to update the security tools, then the computer system may be infected by virus or may be attacked. Through this paper a learning system is being proposed to provide security by identify the operating system process as Self and Non-Self. Concepts of Artificial Neural Network (ANN) Learning have been used for the identification of processes. Initially, an Artificial Neural Network is created by using processes parameters with random weights. These weights are updated by using Gradient Descent Algorithm for various training examples, and then this Artificial Neural Network is tested with test data examples. It has been observed that the Artificial Neural Network Learning provides a better approach for identifying Self and Non-Self process and provides a better security.

Keywords: Self and Non Self Process, Machine Learning, Artificial Neural Network, Gradient Descent, Perceptron

1. Introduction

Information security, Cyber Security and Computer Security [4] are a vital issue. To provide the highest possible extent of computer security, implementation of an efficient and secure operating system has become a necessity [5]. Some operating system developers design a secure operating system and security tools which works to identify the unauthorized access of the system. Many software and hardware based computer security tools are developed by various vendors as Computer Security Tool [6]. Most of the operating systems cannot provide the highest level of security due to deign constraints. To achieve the maximum security for a computer system, a major change in the operating system is required. With the help of this paper a new methodology is proposed to provide the maximum security by identification of Self and Non-Self process [1,2,3] using concepts of Machine Learning [7,9,10]. The operating system processes generated by virus, worms etc can be classified as Non-Self. The operating system processes generated for system software or by some reliable application software can be classified a Self. In this paper, the main concept is to identify these Non-Self processes with the help of an Artificial Neural Network.

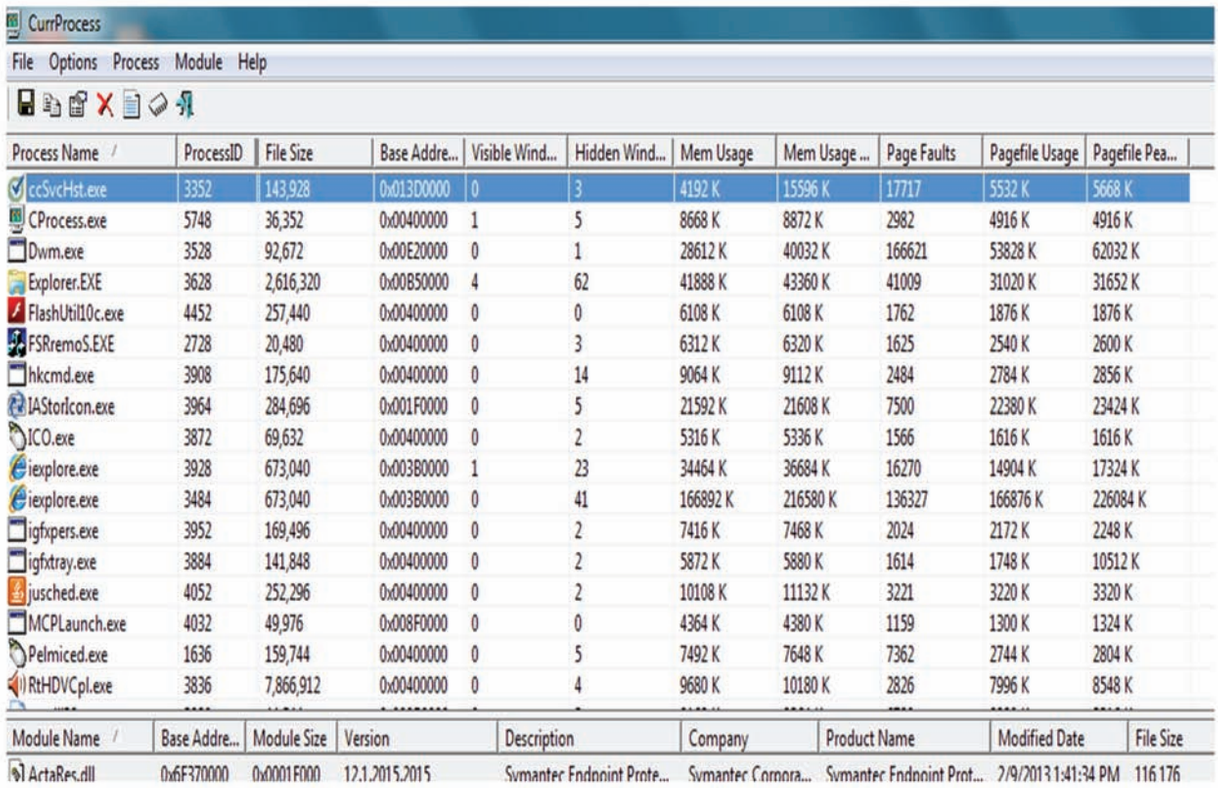
Machine Learning is a rapidly growing field of artificial intelligence. Tom M. Mitchell [7] has provided a formal definition: "A computer program is said to learn from experience E with respect to any class of tasks T and performance measure P, if its performance of tasks in T, as measured by P, improves with experience E". Machine Learning deals with the development of such computer programs which automatically improves their performance and gain experience. These concepts can be used to provide better security. Artificial Neural Network has been used in this paper to implement the concept of Machine Learning.

Various Machine Learning concepts like Instance-Based Learning, Decision Tree Learning, Bayesian Learning, learning through Artificial Neural Network (ANN),

* Assistant Professor, Department of Computer Science and Engineering, Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India. E-mail: amitrathi10@yahoo.co.in

** Professor, Department of Computer Science and Engineering, Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India. E-mail: dr.shishir@yahoo.com

Figure 1: Screenshot of CurrProcess Tool



Concept Learning, Genetic Algorithm, Analytical Learning [7] etc. In this paper a methodology is being proposed, in which learning will be provided through Artificial Neural Network.

ANN learning [7,11,12,13] provides a scheme for learning real-valued, discrete valued and vector-valued functions from a given set of examples. ANN learning is motivated by the Natural Learning Systems. Gradient Descent algorithm [14,15,16] is generally used in ANN for learning. ANN learning is robust in the training data and has been successfully applied to problems such as interpreting visual scenes, learning robot strategies, face recognition, handwritten character recognition, speech recognition etc. In the proposed approach, this algorithm has been applied for creating Perceptron to identify the operating system processes as Self and Non-Self.

2. Proposed Methodology

In a computer system all the viruses and attacks create its own processes in the system. Proposed methodology works on the processes and its parameters to identify the process generated by viruses or attacks. These processes will be identified as Non-Self by using the concepts of ANN.

A process has many attributes like Process ID, Priority, Product name, Version, Description, Company, Window Title, File size, File Created Date, File Modified Date, File Name, Base Address, Created On, Visible Windows, Hidden Windows, User Name, Memory Usage, Memory Usage Peak, Page Faults, Pagefile Usage, Pagefile Peak Usage and File Attributes. Initially for the proposed approach, those parameters are selected which does not have null values. By using the Curprocess tool [8] initially five attributes Process ID, File size, Memory Peak Usage, Page Faults and Page File Peak Usage will be used in the proposed approach. Figure 1 shows the screen shoot of CurrProcess tool window.

3. Range of the Parameters

Every process parameter has some values like a number, size of the file in bytes or KB, address in hexadecimal format, character etc. These values have some lower and higher range as shown in Table 1. By using Currprocess tool on various system conditions (work load), process parameters values are identified. Table 1 shows minimum and maximum range parameters.

Table 1: Process Parameters and its Minimum and Maximum Range.

Parameter	Range Min - Max
Process ID	000-9999
File Size	00000 – 9999999 (Bytes)
Base Address	0x00000000 – 0x99900000
Hidden Windows	0 -999
Memory Usage	000 – 999999 (K)
Memory Peak Usage	000 – 999999 (K)
Page Faults	0000 – 9999999
Page File Usage	000 – 999999 (K)
Page File Peak Usage	000 – 999999 (K)

4. Range Selected for Learning

Initially for the proposed approach, five process parameters are used to identify the Self and Non-Self processes. Better security can be achieved by using more parameters and dividing parameter ranges into small parts. After analyzing various processes initially five parameters: - Process ID (divided into three ranges: - low, medium and high), File Size (divided into three ranges: - low, medium and high), Memory Peak Usage (divided into five ranges: - very low, low, medium, high and very high), Page Fault (divided into five ranges: - very low, low, medium, high and very high), Page File Peak Usage (divided into three ranges: - low, medium and high) are used for ANN. Minimum and minimum ranges of these selected parameters have been divided as follows.

Range of Process ID has been divided into three parts:-

Low – 1938 and Below
Medium – 1939 to 3163
High – 3164 and Above

Range of File Size range has been divided into three parts:-

Low – 314688 and Below
Medium – 314689 to 4375625
High – 4375626 and Above

Range of Memory Peak Usage has been divided into five parts:-

Very Low – 10490 and below
Low – 10491 to 31302
Medium – 31303 to 78172
High – 78173 to 109391
Very High – 109392 and above

Range of Page Faults has been divided into five parts:-

Very Low – 2274 and below
Low – 2275 to 5358
Medium – 5359 to 25001
High – 25002 to 43750
Very High – 43751 and above

Range of Page File Peak Usage has been divided into three parts:-

Low – 5008 and below
Medium – 5009 to 31269
High – 31270 and above

The range of these process parameters can be changed according to the system architecture & organization and operating system running on the computer system.

5. Training Examples

For application of any Learning Methods in Machine Learning, sets of training examples have been used. Training example, plays an important role to provide learning to a system. These sets have both positive and negative examples as shown in Table 2. There are 25 training examples in Table 2, in which 18 are positive (Self) and 7 are negative (Non-Self). These examples are used to provide the training to ANN. The values of parameters are converted as per the above section 4 into Very Low (VL), Low (L), Medium (M), High (H) and very High (VH) to make the easy calculation and understanding. The abbreviations VL, L, M, H and VH are used instead of actual values. These training examples are selected after various running conditions on various workloads of a computer system. The system was virus infected during these observations. During these conditions 25 different processes are identified as training examples as shown in Table 2. In these 25 training examples, eighteen examples are positive examples defined as Self (system and some application processes) and seven examples are negative examples defined as Non-Self (generated by viruses).

6. Neural Network Learning

Neural Network Learning methods give a forceful approach to approximating discrete-valued, real-valued,

Table 2: Training Example Set

	ProcessID	*File Size (KB)*	*Mem Usage Peak (KB)*	*Page Faults*	*Pagefile Peak Usage (KB)*	*Self*
P1	L	M	M	VH	H	Yes
P2	H	M	VL	L	M	Yes
P3	M	L	VL	VL	L	Yes
P4	H	M	M	M	M	No
P5	M	M	L	M	M	Yes
P6	H	L	L	M	M	No
P7	L	M	L	L	M	Yes
P8	H	L	H	VH	H	Yes
P9	H	M	VL	VL	L	No
P10	H	L	L	L	M	Yes
P11	L	M	M	H	H	Yes
P12	M	L	VH	VH	H	Yes
P13	H	M	VL	L	M	No
P14	H	H	M	M	H	Yes
P15	H	M	L	M	M	Yes
P16	H	H	M	M	H	No
P17	M	L	L	VH	M	Yes
P18	L	L	VL	VL	L	Yes
P19	H	H	VL	L	M	No
P20	H	M	L	L	M	No
P21	H	M	M	M	M	Yes
P22	M	M	L	M	M	Yes
P23	M	H	M	M	M	Yes
P24	M	M	M	M	M	Yes
P25	M	H	M	M	M	Yes

and vector-valued target function. An Artificial Neural Network system is based on a component; called a Perceptron [17, 18]. Perceptron takes a vector of real-valued inputs; compute a linear combination of these inputs, then outputs a 1 if the result is larger than some threshold and -1 otherwise. Figure 2 shows the Neural Network for identification of Self and Non-Self process, which has five inputs as according to five processes parameters used in the proposed approach. Number of inputs and hidden layers may be changed if more processes parameters are considered.

Learning a Perceptron involves choosing and updated the values for the weights W_0,W_1….. Wn. For learning, the Perceptron begins with random weights (or some unit or same weight) and then iteratively apply the Perceptron to each example, modifying the Perceptron weights. This process is repeated, iterating through the training examples as many times as needed until the Perceptron classifies all training examples correctly. Weights are modified at each step according to the Perceptron training rule by using the Gradient Descent algorithm [7,14,15,16] as mentioned below-

Gradient-Descent (*training_example,* η)

Every training example is a pair of the form <X,t>, where X is the vector of input values, and t is the target output value. H is the learning rate (e.g. 0.5).

- Initialize each Wi to various small random values (initially can be taken as 1).
- Until the termination state is met, Do
  - Initialize each ΔWi to zero.
  - For each <X, t> in *training_example*, Do

- Input the instance X to the unit and calculate the output o
- For each linear unit weight Wi , Do

$$\Delta Wi = \eta (t - o) Xi$$

  - For each linear unit weight Wi, Do

$$Wi \leftarrow Wi + \Delta Wi$$

In the above learning algorithm "t" is the target output for the present example, "o" is the output generated by the Perceptron, and "η" is a positive constant called the learning rate. The job of the learning rate is to moderate the degrees to which weights are altered at each step. If the training example is properly classified already by Perceptron, in this case, (t-o) is zero, making "ΔWi" zero, so that no weights are updated. Suppose the Perceptron output "-1" when the target output is "+1". To make the Perceptron output "+1" instead "-1" in this case, the weights must be altered to increase the value of "WiXi". For example, if "Xi > 0", then increase "Wi" will bring the Perceptron closer to correctly classifying this example. In this case the training rule will increase "Wi", because "(t-o)", "η", and "Xi" are all positive. For example, if "Xi = 0.8", "η = 0.1", "t = 1", and "o = -1", then the weight update will be –

$$\Delta Wi = \eta (t - o)Xi$$
$$= 0.1 (1 - (-1))0.8$$
$$= 0.16.$$

On the other hand, if "t = -1" and "o = 1", then weights associated with positive "Xi" will be decreased rather than increased.

For hidden layer calculation of ANN, the values of process parameters have been taken as the mean of the range during the execution of the Gradient Descent algorithm. The values use for the hidden layer calculation is as below-

Value of Process ID/DID has been used as-

Low (L) – 1938 and Below Value used in Neural Network(NN) = (0+1938)/2 = 969

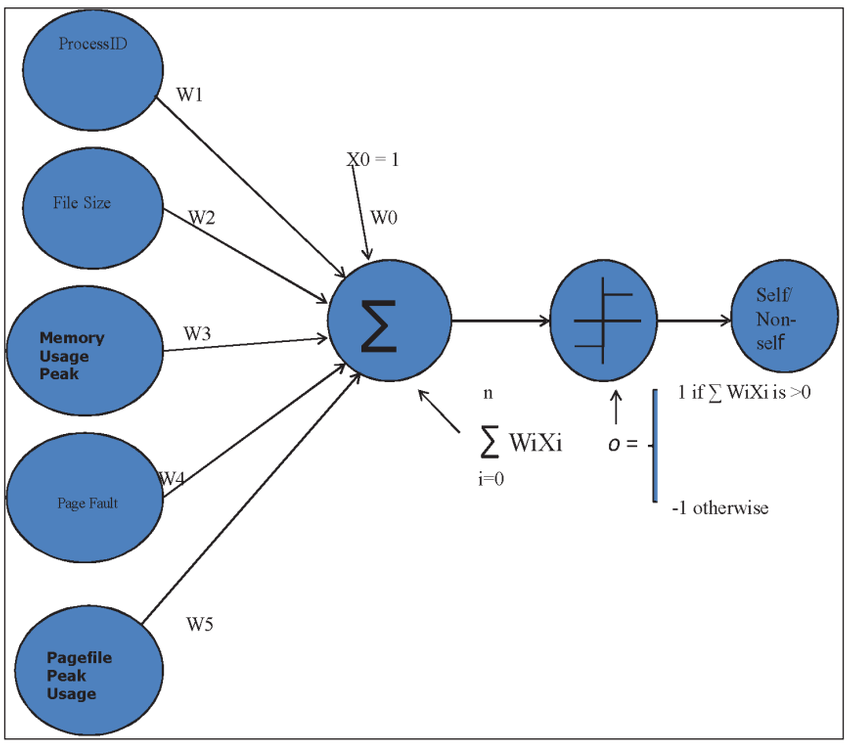
Medium – 1939 to 3163 Value used in NN = (1939+ (3163-1939)/2) = 2554

High – 3164 and Above Value used in NN = (3164+ (9999-3164)/2) = 6581

Value of File Size /DFS has been used as -

Low – 314688 and Below Value used in NN = (0+314688)/2 = 157344

Figure 2: Neural Network for Identification of Self and Non-Self Processes



Medium – 314689 to 4375625 Value used in NN= (314689+(4375625-314689)/2) = 2345157

High – 4375626 and Above Value used in NN= (435626+(9999999-4375626)/2)= 3247812

Value of Memory Peak Usage / DMPU has used as -

Very Low – 10490 and below Value used in NN = (0+10490)/2 = 5245

Low – 10491 to 31302 Value used in NN= (10491+(31302-10491)/2) = 20897

Medium – 31303 to 78172 Value used in NN= (31303+(78172-31303)/2) = 2345157

High – 78173 to 109391 Value used in NN= (78173+(109391-78173)/2) = 93782

Very High – 109392 and above Value used in NN= (109392+(999999-109392)/2)= 554696

Value of Page Faults /DPF has been used as -

Very Low – 2274 and below Value used in NN = (0+2274)/2 = 1137

Low – 2275 to 5358 Value used in NN= (2275+(5358-2275)/2) = 3817

Medium – 5359 to 25001 Value used in NN= (5359+(25001-5359)/2) = 15180

High – 25002 to 43750 Value used in NN= (25002+(43750-25002)/2) = 34376

Very High – 43751 and above Value used in NN= (43751+(99999-43751)/2)= 71875

Value of Page File Peak Usage /DPFPU has been used -

Low – 5008 and below Value used in NN = (0+5008)/2 = 2504

Medium – 5009 to 31269 Value used in NN= (5009+(31269-5009)/2) = 18139

High – 31270 and above Value used in NN= (31270+(99999-31270)/2)= 65635

From the training example in Table 2, processes parameter values are used as according to the above classification, e.g. If the value of Page Fault is High (H) then for neural network calculation is 34376. The Gradient Descent algorithm has been applied to the training data of Table 2. The values of process parameters are scaled down according to the desired output or the final output is scaled on 1 to -1 range. After several iterations of Gradient Descent, every weight is/weights are updated to a proper required value. The weights are updated during each iteration, when there will be no updates in the weights, then further iteration of the algorithm is not required. If a separate training data is also used, then these updated weights are slightly changed according

Table 3: Test Data Set

	ProcessID	*File Size (KB)*	*Mem Usage Peak (KB)*	*Page Faults*	*Pagefile Peak Usage (KB)*	*Self*
P1	M	M	M	M	M	Yes
P2	M	L	H	H	H	Yes
P3	H	L	VL	VL	L	Yes
P4	M	M	L	L	M	No
P5	M	L	VL	M	M	Yes
P6	L	M	L	M	M	Yes
P7	H	M	M	VH	H	Yes
P8	L	M	L	M	M	Yes
P9	M	H	L	L	M	No
P10	L	L	L	M	M	No
P11	H	H	L	M	M	Yes
P12	H	M	L	M	M	Yes
P13	H	M	VL	L	M	Yes
P14	H	H	M	M	H	Yes
P15	M	M	VL	VL	L	Yes

to new training data set. Various training sets have been used to find out the appropriate weight. When there will be no update in weights and ANN classify all training examples correctly, then this ANN will be fully learned.

After providing training to ANN, this trained ANN is tested with a test data as shown in Table 3. After providing training by training data of Table 2 (after first iteration) it is then tested by the test data of Table 3, it has been found that the ANN classify P6, P10, and P14 incorrectly. When the number of iteration increases near about 90 then ANN classifies all test data correctly.

Weight update of one parameter during iteration may affect the update of other parameter's weight, so the weights are updated slightly, not by a large value. For the same a error function [7] is used to maintain the processes of weight update.

$$E_d(w) = \tfrac{1}{2} (t_d - o_d)^2$$

Where t_d and o_d are the target value and the unit output value for training example d. Iteration of the algorithm over the training example d in D, at each iteration altering the weights according to the gradient with respect to $E_d(w)$.

7. Experimental Results and Comparisons

As the iteration of the Gradient Descent algorithm increases and weights are updated, error in observed and desired output decreases as shown in the Figure 3 and Figure 4. Weight update of all input to hidden layer are updates slightly in the direction of positive or negative side as the number of iteration increases as shown in Figure 5. So, it will be decided when to stop more iteration i.e. How many iterations are sufficient to provide a decision on Self and Non-Self processes. As it has been cleared from the graph of Figure 3, 4 & 5; that 400 to 500 iterations are sufficient in this case.

It has been observed from the Figure 6, as the numbers of input nodes (parameters) in the ANN increases, the security also increases. It is clear from the graph of Figure 6, that when the nodes increases after fifteen, then the security remain nearly about constant. It has been observed from the Figure 7, as the number of input nodes (parameters) in the Artificial Neural Network, there is a degradation in system performance. It is clear from the graph of Figure 7, that when the nodes increases after 10, then the system performance decreases very rapidly. So, to construct an ANN to find out Non-Self process, all of the above results play a very important role.

Figure 3 **Plots of Error E as a Function of the Number of Weight Updates**

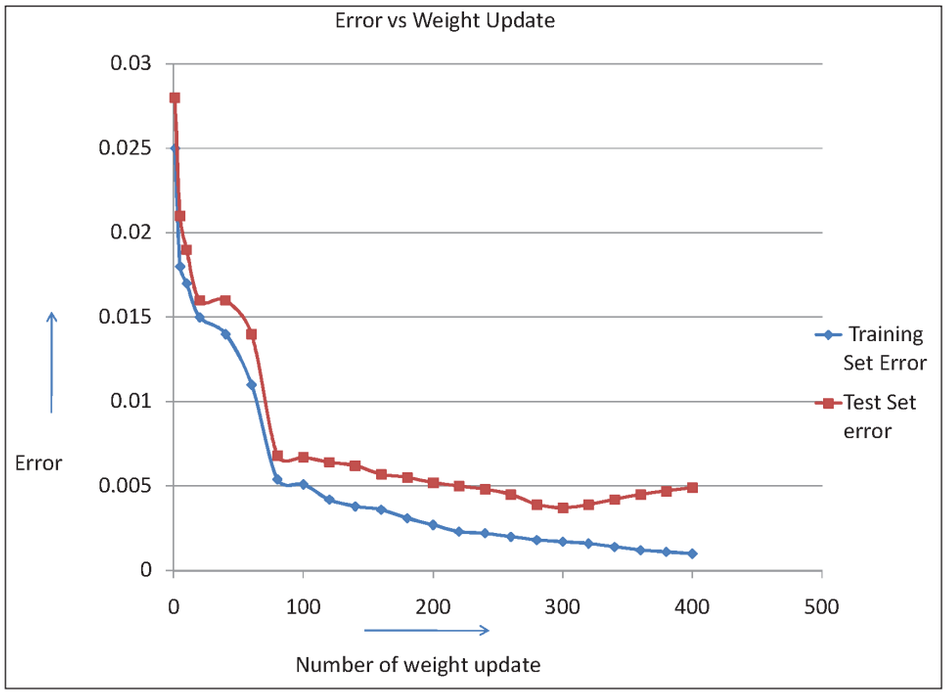


Figure 4: Plots of Output Error as the Number of Iteration Increases

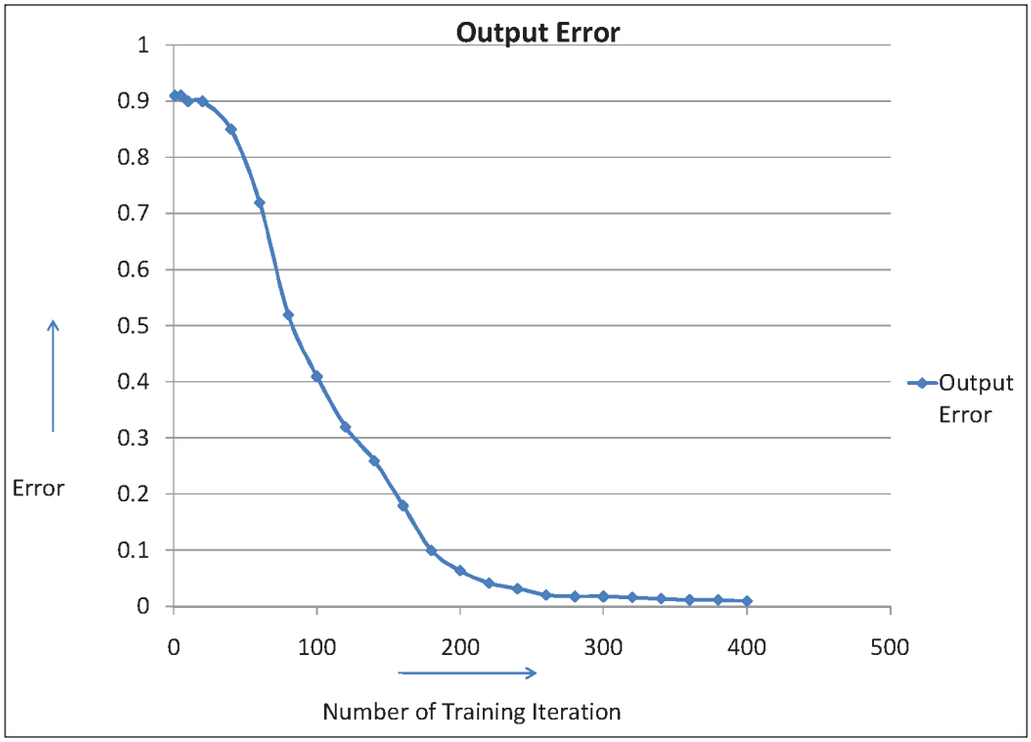


Figure 5: Plots of Weight Update from Input to Hidden Layer

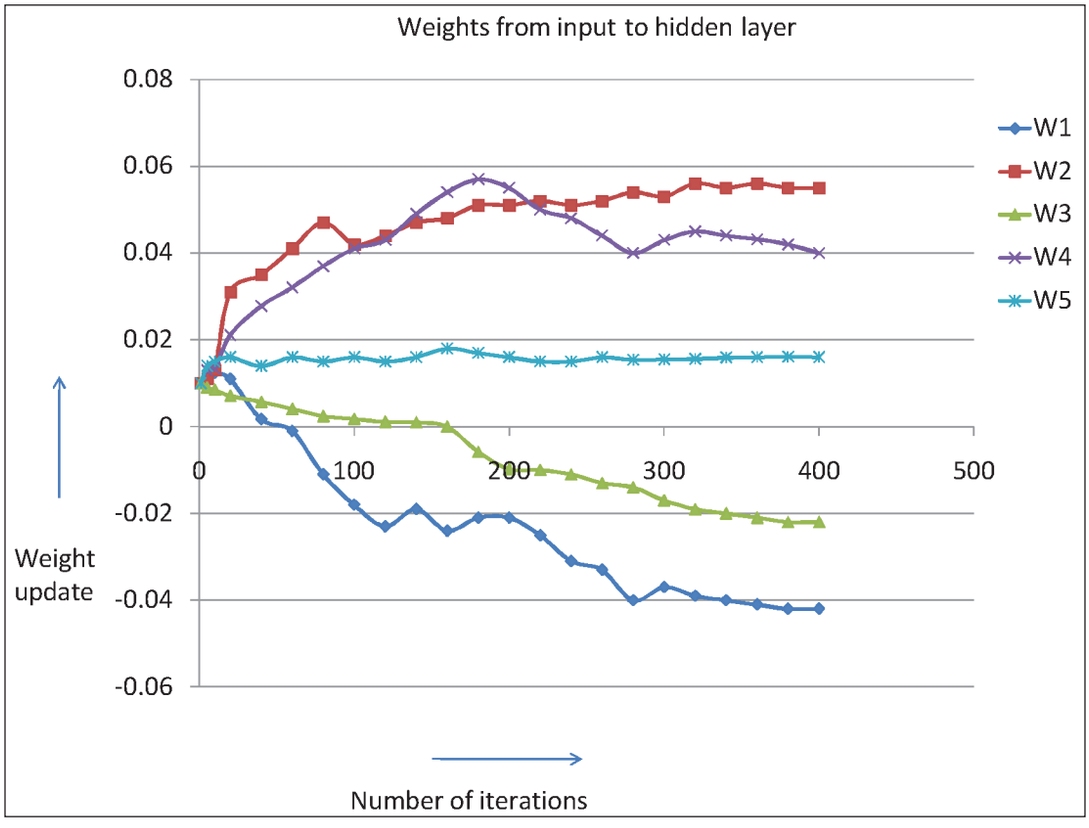


Figure 6: Graph between Number of Input Nodes and Security Level

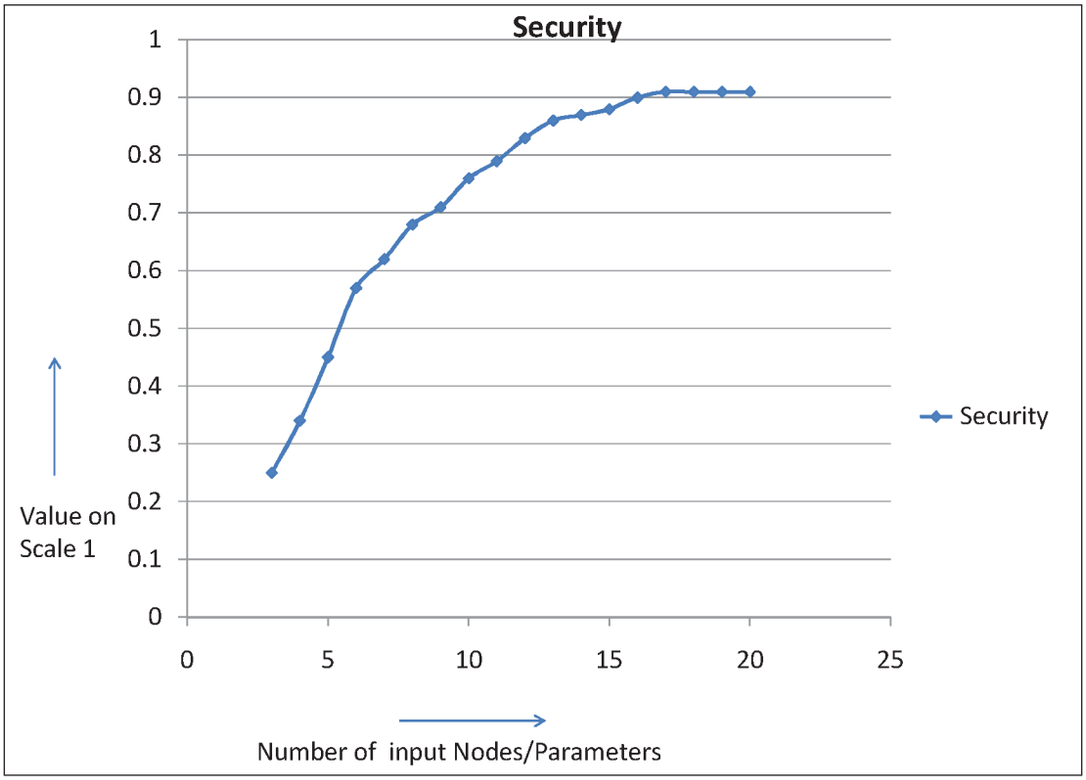


Figure 7: Graph between Number of Input Nodes and Decrease in System Performance

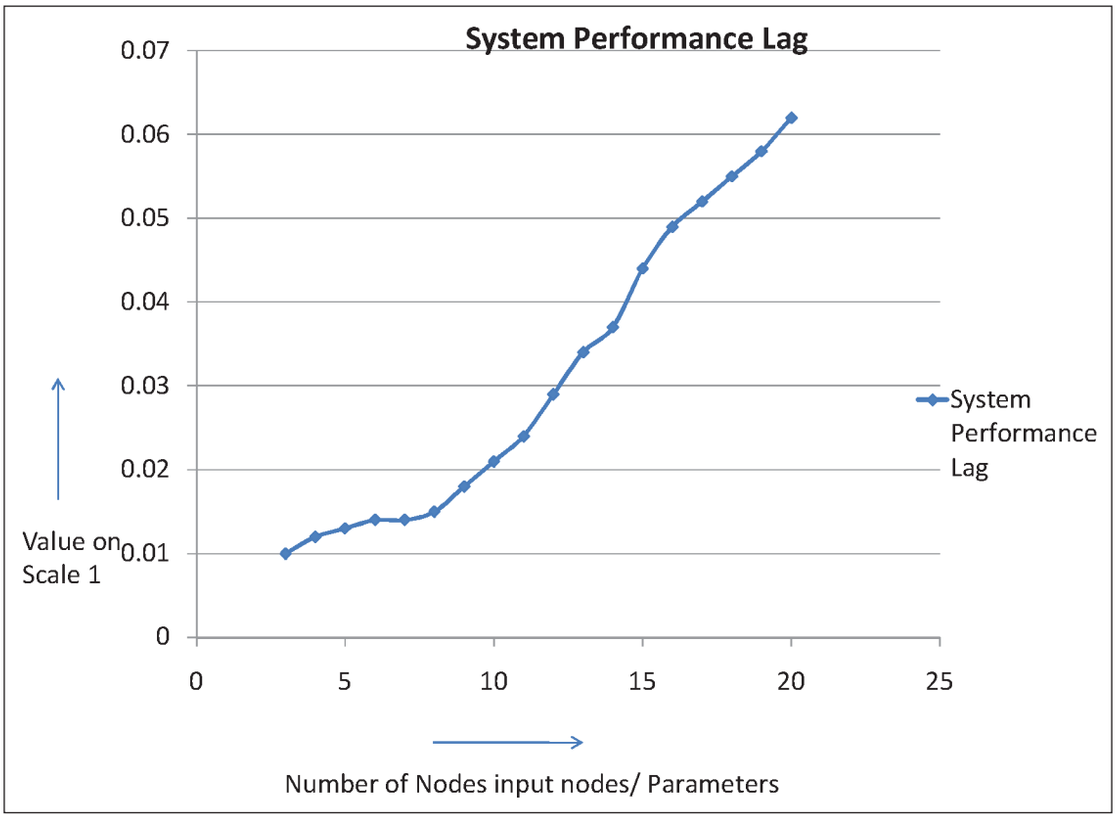


Table 4: Comparison of Proposed Approach with Some Anti-virus tools

Parameters / Anti-Virus	Scan Time	Accuracy	Detection Rate	Performance Lag	Signature based Detection	Regular Updating Required
AVG Anti-virus	High	High	Normal	Yes	Yes	Yes
Norton Anti-virus	High	High	High	Yes	Yes	Yes
Avast Anti-virus	High	Medium	Normal	Yes	Yes	Yes
Microsoft Security Essentials	Average	High	High	Yes	Yes	Yes
Proposed Approach	Low	Very High	Very High	Yes	No	No

The proposed ANN approach is better than the existing security approach as shown in Table 4. The proposed ANN approach takes very less time to find out the Non-Self processes, as other approaches scan all the files and data. Accuracy and detection rate is very high in comparison to existing anti-virus tools, proposed approach scan only the processes. As the proposed ANN approach scans all processes, so the system will become slow. The proposed ANN approach is free from signature as required in existing approaches, proposed approach works on parameter's value not on any signature. No regular update is required in the proposed approach as it is required in anti-virus tools; to generate new detectors existing detector set is used.

8. Conclusions and Future Work

Various approaches and tools are used to provide security, but these approaches are not sufficient to provide best security level. Artificial Immune approach plays an important role to provide better security to the computer system. The Artificial Neural Network approach used in this paper provides better result over the current approach. In these approaches, Artificial Neural Network provides the best result to provide a learning system. By adjusting the value of learning rate η used in the Gradient Descent algorithm to get the better learn system. By using many training and test data set, a better weight update can be adjusted and it will provide a better result.

References

1. Percus, J. K., Percus, O. E., & Perelson, A. S. (1992). Probability of self-nonself discrimination. *Theoretical and Experimental Insights into Immunology*, 66, 63-70.
2. Forrest, S., Hofmeyr, S. A., Somayaji, A. B., & Longstaff, T. A. (1996). *A Sense of Self for UNIX Processes*. Proceedings of IEEE Symposium on Computer Security and Privacy. Retrieved from http://www.cs.unm.edu/~immsec/publications/ieee-sp-96-unix.pdf
3. Forrest, S., & Perelson, A. S. (1994). *Self Non-Self Discrimination in a Computer*. In Proceedings of the IEEE Symposium on Research in Security and Privacy. Retrieved from http://www.cs.unm.edu/~immsec/publications/virus.pdf
4. Solms, R. V., & Niekerk, J. V. (2013). From information security to cyber security. *Elsevier's Computer & Security*, October, 38, 97-102.
5. Yang, C. Q. (2003). *Operating System Security and Secure Operating Systems*. Global Information Assurance Certification Paper. Retrieved from http://www.giac.org/paper/gsec/2776/operating-system-security-secure-operating-systems/104723
6. http://www.cyberwarzone.com/massive-cyber-security-tools-list-2013
7. Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill International Editions, Computer Science Series.
8. http://www.nirsoft.net/utils/cprocess.html
9. Haoyong, L., & Tang, H. (2011). *Machine Learning Methods and their Application Research*. IEEE International Symposium on Intelligence Information Processing and Trusted Computing, (pp. 108-110).
10. Hua, W., Cuiqin, M. A., & Lijuan, Z. (2009). *A Brief Review of Machine Learning and its Application*. IEEE Information Engineering and Computer Science (pp. 1-4).
11. Nguyen, D. H., & Widrow, B. (1990). *Neural Networks for Self-Learning Control System*. IEEE Control System Magazine (pp. 18-23).
12. Zhang, G. P. (2000). *Neural Networks for Classification: A Survey*. IEEE Transaction on System, Man and Cybernetics-Part C: Applications and Reviews, 30(4), 451-462.

13. Baesens, B., & Bouboulis, P. (2012). *Neural Networks and Learning Systems Come Together*. IEEE Transactions on Neural Networks, 23(1), 1-6.
14. Mandic, D. P. (2004). *A Generalized Normalized Gradient Descent Algorithm*. IEEE Signal Processing Letters, 11(2), 155-118.
15. Ahmad, F., & Isa, N. A. M. (2010). *Performance Comparison of Gradient Descent and Genetic Algorithm Based Artificial Neural Networks Training*. 10th International Conference on Intelligent Systems Design and Applications (pp. 604-609).
16. Xu, D., Li, Z., Wu, W., Ding, X., & Qu, D. (2007). *Convergence of Gradient Descent Algorithm for Diagonal Recurrent Neural Networks, Bio-Inspired Computing: Theories and Applications*, (pp. 29-31).
17. Watterson, J. W. (1990). *An Optimum Multilayer Perceptron Neural Receiver for Signal Detection*. IEEE Transactions on Neural Network, 1(4), 280-300.
18. Pal, S. K., & Mitra, S. (1992). *Multilayer Perceptron, Fuzzy Sets and Classification*. IEEE Transaction on Neural Networks, 3(5), 683-697.