An Improved Neural Network Based Approach For Identification of self & Non-Self Processes

AMIT KUMAR, SHISHIR KUMAR
Department of Computer Science and Engineering
Jaypee University of Engineering and Technology
A.B. Road, Raghogarh, Guna
INDIA
amitrathi10@yahoo.co.in http://www.juet.ac.in/Department/faculty.php?id=42483894&dep=cse,
dr.shishir@yahoo.com http://www.juet.ac.in/Department/faculty.php?id=45652778&dep=cse

Abstract:-Security of computer systems is a very crucial issue. Now a days various security approaches and tools are used to protect computer system by any virus, worms and attacks. These security tools require a regular signature based updating to protect the computer system by latest virus and worms. If a system has not updated its security tool, then it may be infected by any virus or worms, then the operating system generates its processes. These processes are harmful to the computer system. These processes are categorized as non-self processes. In this paper an Artificial Neural Network is designed to identify the self and non-self operating system process. Backpropagation Algorithm is used to provide the training and learning to the Artificial Neural Network. Initially an Artificial Neural Network is created with random input weights. These weights are updated by using Backpropagation Algorithm for various training examples. After the weight update Artificial Neural Network tests by various test data examples. After campaigning with various computer security approaches it has been observed out, that Artificial Neural Network provides a better security by identifying self and non-self process.

Key words:-Security, Self Process, Non Self Process, Backpropagation, Weight, Activation Function.

1 Introduction

Computer Security[1] is a very critical issue. Various terms can be used instead of computer security like information security, Cyber Security, etc. To provide the highest level of computer security, operating system developers required to implement an efficient and secure operating system[2]. Many OS developers, design secure operating systems and some security tools. The objective of these OS a security tools are to identify the unauthorized access of the system. Many software and hardware based security tools are also available for the computer designed by various vendors[3]. Lots of operating systems cannot provide the best computer security level due to its design limitations. To get the maximum security level for a computer system, lots of major change is required in the design of an operating system. By using the concept of Artificial Neural Network we proposed a methodology to provide the maximum security by identification of self and non-self process[4,5 6].

The Artificial Neural Network is a major conceptual concept of machine learning[7,8,9]. The operating system processes can be categorized into two parts self and non-self. The processes which are generated by system software, application software or any trusted software can be classified as non-self. The processes which are generated by viruses, worms can be classified as non-self. In this paper the an Artificial Neural Network is designed to identify these non-self processes.

The concepts of Machine learning are used to provide better security. Tom M. Mitchell[7] provided a formal definition of Machine Learning: "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 in the Artificial Neural Network approach to provide better security.

There are lots of Machine Learning concepts like Concept Learning, Decision Tree Learning, learning through Artificial Neural Network (ANN), Bayesian Learning, Instance-Based 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.

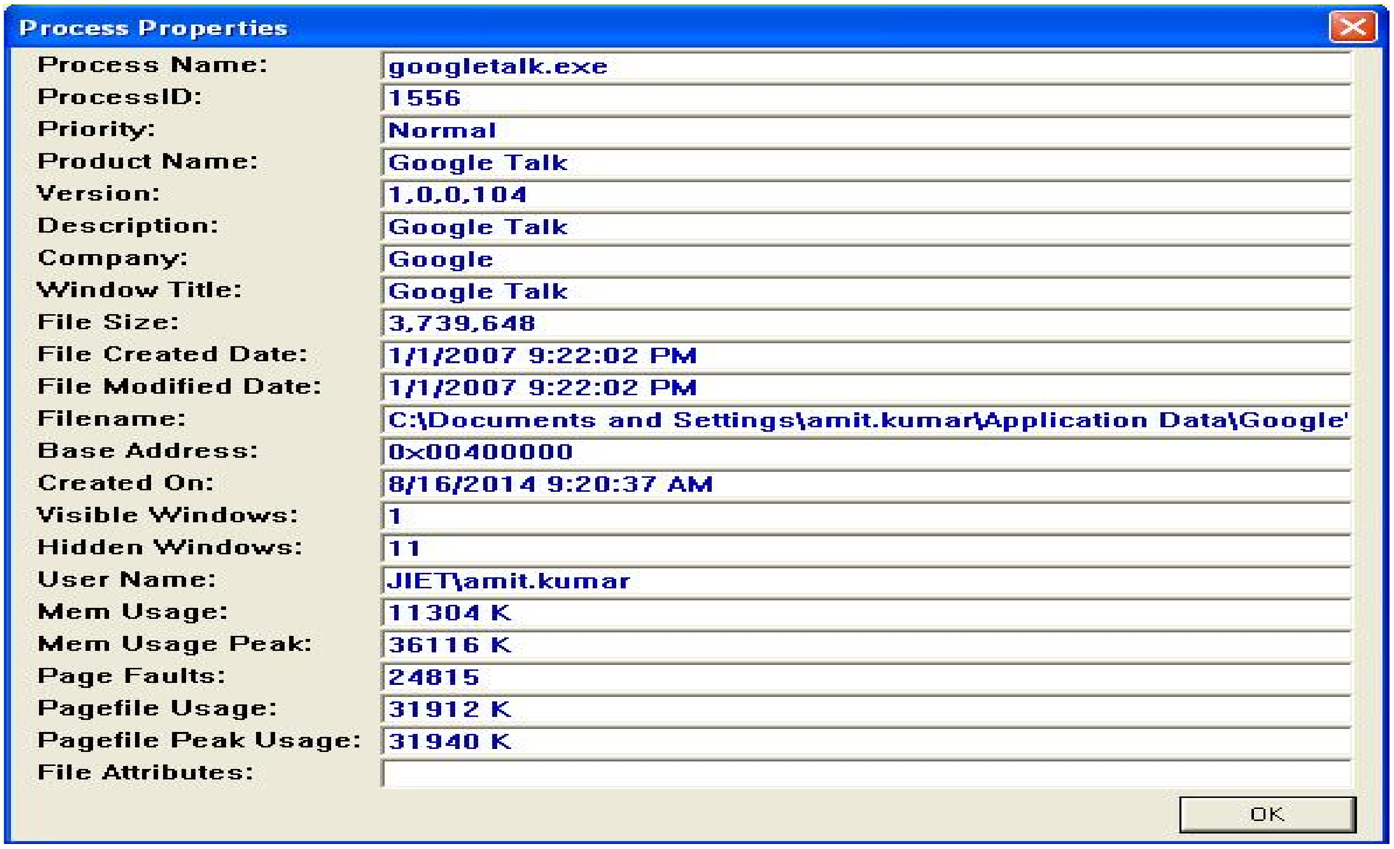
Artificial Neural Network learning[7, 10,11,12] can be implemented on real-valued, discrete valued, and vector-valued functions from a given set of examples. ANN is motivated by natural neuron learning systems of living things. Backpropagation algorithm[13,14,15] is mostly used in ANN for learning purpose. ANN learning has been successfully applied to various problems like handwritten character recognition, interpreting visual scenes, face recognition, learning robot strategies, speech recognition, etc. In this paper ANN learning approach and Backpropagation algorithm has been applied for to identify the self and non-self process of a computer operating system.

2 Proposed **Methodology**

Operating system of a computer system generates processes of the all programs and software to execute it. If viruses, worms and attacks are not identified by the security tolls then OS of the computer system also creates the processes of these in the system. Proposed approach based on ANN, 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.

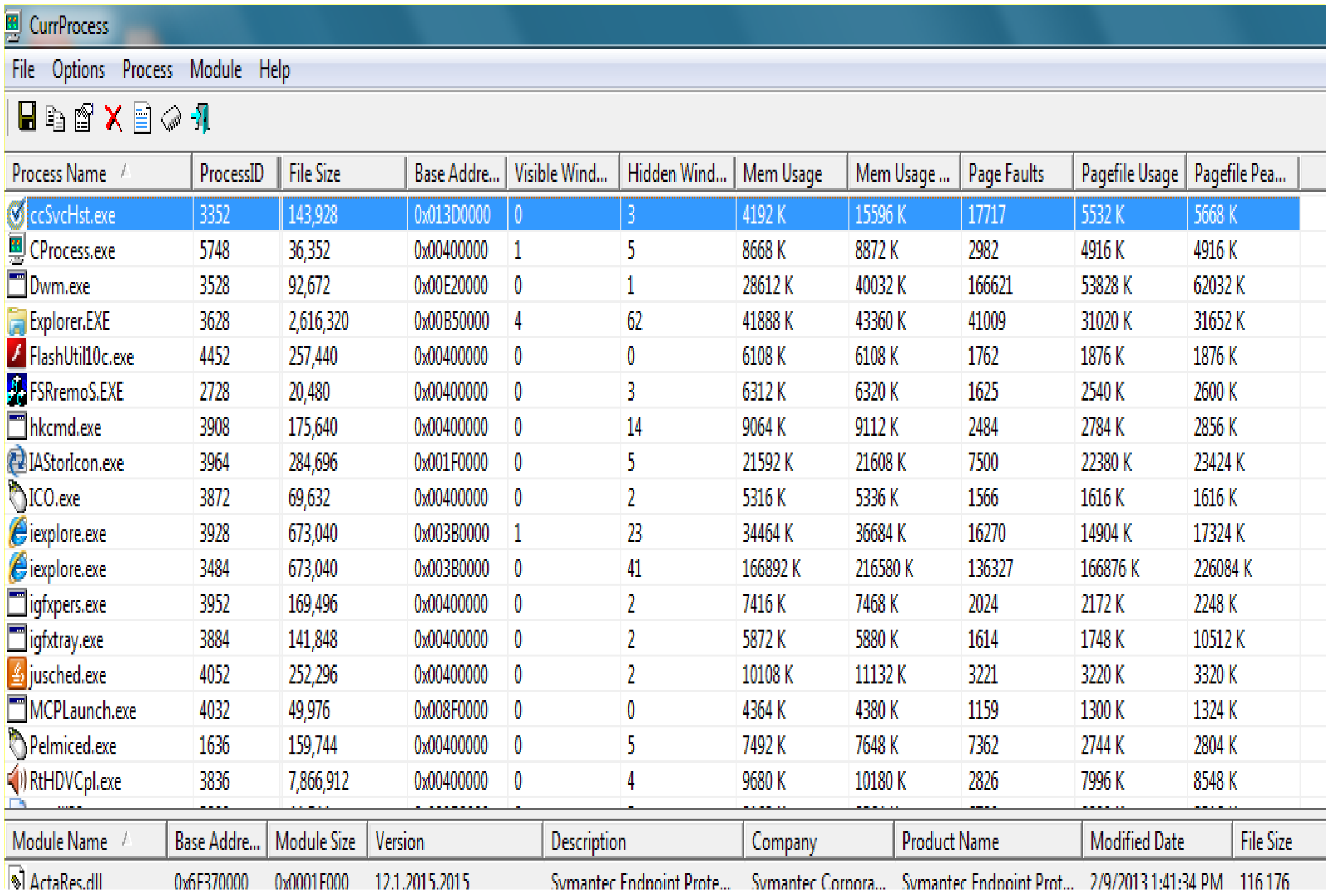
A process during the execution uses its various attributes to complete its task, like:

- *ProcessID* - this shows the identity number of the process
- *Priority* - show the priority to get the CPU time among the other processes the priority may be normal, below normal or above normal.
- *Product name* - show the name of the software Version - show the version number of the software
- *Description* - show the description of the process
- Company - show the name of the company who developed the software, this may be null.
- *Window Title* - If the processes have a GUI, then there will be a Window title for it, , this may be null.
- *File size* – it shows the process's software file size in bytes
- *File Created Date* - it shows the date and time of creation of process's software
- *File Modified Date* - it shows the date and time of modification (if there is any) of the process's software. If there is no modification in the software then the file modification date is same as file created date.
- *File Name* - it shows the path and file name of the executable file.
- *Base Address* - it shows the memory address of the process in main memory.
- *Created On* - it shows the date and time of creation of process in the computer system.
- *Visible Windows* - If the processes have GUI, then there will be some (number) visible Windows for it
- *Hidden Windows* – it shows the number of hidden windows of the process.
- *User Name* - it shows the user name (owner) of the processes.
- *Memory Usage* - it shows the average memory usage in KB of the process during the
- *Memory Usage Peak* - it shows the peak memory usage in KB of the process during the execution.
- *Page Faults* - it shows the number of page faults made by the process during the execution.
- *Pagefile Usage* - it shows the average page file usage in KB of the process during the execution.
- *Pagefile Peak Usage* - it shows the peak page file usage in KB of the process during the execution.



Process Name:	googletalk.exe
ProcessID:	1556
Priority:	Normal
Product Name:	Google Talk
Version:	1,0,0,104
Description:	Google Talk
Company:	Google
Window Title:	Google Talk
File Size:	3,739,648
File Created Date:	1/1/2007 9:22:02 PM
File Modified Date:	1/1/2007 9:22:02 PM
Filename:	C:\Documents and Settings\amit.kumar\Application Data\Google'
Base Address:	0x00400000
Created On:	8/16/2014 9:20:37 AM
Visible Windows:	1
Hidden Windows:	11
User Name:	JIET\amit.kumar
Mem Usage:	11304 K
Mem Usage Peak:	36116 K
Page Faults:	24815
Pagefile Usage:	31912 K
Pagefile Peak Usage:	31940 K
File Attributes:	

Figure 1: Process Properties of googletalk.exe



Process Name	ProcessID	File Size	Base Addre...	Visible Wind...	Hidden Wind...	Mem Usage	Mem Usage ...	Page Faults	Pagefile Usage	Pagefile Pea...
ccSvcHst.exe	3352	143,928	0x013D0000	0	3	4192 K	15596 K	17717	5532 K	5668 K
CProcess.exe	5748	36,352	0x00400000	1	5	8668 K	8872 K	2982	4916 K	4916 K
Dwm.exe	3528	92,672	0x00E20000	0	1	28612 K	40032 K	166621	53828 K	62032 K
Explorer.EXE	3628	2,616,320	0x00B50000	4	62	41888 K	43360 K	41009	31020 K	31652 K
FlashUtil10c.exe	4452	257,440	0x00400000	0	0	6108 K	6108 K	1762	1876 K	1876 K
FSRremoS.EXE	2728	20,480	0x00400000	0	3	6312 K	6320 K	1625	2540 K	2600 K
hkcmd.exe	3908	175,640	0x00400000	0	14	9064 K	9112 K	2484	2784 K	2856 K
IAStorIcon.exe	3964	284,696	0x001F0000	0	5	21592 K	21608 K	7500	22380 K	23424 K
ICO.exe	3872	69,632	0x00400000	0	2	5316 K	5336 K	1566	1616 K	1616 K
iexplore.exe	3928	673,040	0x003B0000	1	23	34464 K	36684 K	16270	14904 K	17324 K
iexplore.exe	3484	673,040	0x003B0000	0	41	166892 K	216580 K	136327	166876 K	226084 K
igfxpers.exe	3952	169,496	0x00400000	0	2	7416 K	7468 K	2024	2172 K	2248 K
igfxtray.exe	3884	141,848	0x00400000	0	2	5872 K	5880 K	1614	1748 K	10512 K
jusched.exe	4052	252,296	0x00400000	0	2	10108 K	11132 K	3221	3220 K	3320 K
MCPLaunch.exe	4032	49,976	0x008F0000	0	0	4364 K	4380 K	1159	1300 K	1324 K
Pelmiced.exe	1636	159,744	0x00400000	0	5	7492 K	7648 K	7362	2744 K	2804 K
RtHDVCpl.exe	3836	7,866,912	0x00400000	0	4	9680 K	10180 K	2826	7996 K	8548 K

Module Name	Base Addre...	Module Size	Version	Description	Company	Product Name	Modified Date	File Size
ActaRes.dll	0x6F370000	0x0001F000	12.1.2015.2015	Symantec Endpoint Prote...	Symantec Corpora...	Symantec Endpoint Prot...	2/9/2013 1:41:34 PM	116,176

Figure 2: Screenshot of CurrProcess tool

- *File Attributes*- this field shows the attribute of the file of the process, this may be empty of has the value A or C, this may be null.

Figure 1 shows the properties of the Process of googletalk.exe. Figure 1 shows all the process parameters as described above. For the initial implementation of the ANN we identify those parameters of a process which has not null values. By using the Curprocess tool[16] initially ten attributes Process ID, Priority, File Modified date, Base Address, Number of Visible windows, Number of Hidden windows, File size, Memory Peak Usage, Page Faults and Page File Peak Usage will be used in the proposed approach. Figure 2 shows the screen shoot of CurrProcess tool window.

3 Range of the parameters

Operating system process parameter has some values like a number, size of the file in bytes or KB, address in hexadecimal format, character, etc. It may be possible that some parameters are null (or zero value) like window title, file attributes, Number of Visible windows and Number of Hidden windows. These values of process parameters have some lower and higher range as shown in Table 1. By using the Currprocess tool for various system running conditions minimum and maximum range of the process parameter have been identified as shown in Table 1.

Parameter	Range Min - Max
Process ID	000-9999
Priority	Below Normal, Normal, Above Normal
File Modified date	A date and time range for current date
Base Address	0x00000000 – 0x99900000
Number of Visible windows	0 -9
Number of Hidden Windows	0 -99
File Size	00000 – 9999999 (Bytes)
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)

Table 1: Process Parameters and its minimum and maximum range.

3.1 Range for Learning and Preprocessing

Initially for the proposed approach ten process's parameters are used for the identification of the self and non-self processes. These ten parameters are Process ID, Priority, File Modified Date, Base Address, Number of Visible windows, Number of Hidden Windows, File Size, Memory Peak Usage, Page Fault, and Page File Peak Usage.

The value of Process ID(PID, a real number), File Size(FS, in KB), Memory Peak Usage(MPU, in KB), Page Fault(PF, a real number), are Page File Peak Usage(PFPU, in KB) is in the form of accountable size i.e. in number form. So, there values are converted into the Very Low (VL), Low (L), Medium (M), High (H) and very High (VH). In the training examples the abbreviations VL, L, M, H and VH are used instead of actual values.

The value of the Priority(PR) may be Below Normal(BN), Normal(N) and Above

Normal(AN). The value of the File Modified Date (FMD) is in the form of MM/DD/YYYY.

The value of Base Address (BA) is the form of 0xHHHHHHHH, where H represents a hexadecimal digit, i.e. the base address is in hexadecimal form. Number of Visible Windows (NVW) is a real number, it may be zero also. Number of Hidden Windows (NHW) is a real number, it may be zero also.

For the easy and simple calculation the range of some of these ten parameters is divided into various parts as shown following:

Process ID (PID) range has been divided into five parts –

Very low	- 714 and below
Low	– 715 to 1938
Medium	- 1939 to 3163
High	- 3164 to 4388
Very High	- 4389 and above

File Size (FS) range has been divided into five parts -

Very Low	- 261419 and below
Low	– 261420 to 314688
Medium	- 314689 to 4375625
High	- 4375626 to 8436562
Very High	- 8436563 and above

Memory Peak Usage (MPU) range 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

Page Faults (PF) range 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

Page File Peak Usage (PFPU) range has been divided into three parts -

Very Low	- 2367 and below
Low	– 2368 to 5008
Medium	- 5009 to 31269
High	- 31270 to 57530
Very High	- 57531 and above

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

The value of the Priority(PR), File Modified Date (FMD), Base Address (BA), Number of Visible Windows (NVW), Number of Hidden Windows (NHW) are used as shown below:-

- The value of the Priority(PR) are (used on the scale of 1000) :
  Below Normal(BN) - 250
  Normal(N) - 500
  Above Normal(AN) – 750

- The value of the File Modified Date (FMD) is converted into an age (in days) after subtracting from current date. For example processes P4 in Table 2 has the FMD = 1/1/2007, after subtracting from the current date (8/25/2014) we get the age factor of the file = 2754 days. FMD filed is used as the age of the file in number of days.

- The value of Base Address (BA) is the form of 0xHHHHHHHH, where H represents a hexadecimal digit. The value of Base Address (BA) is used as follows:
  0x00000000 to 0x004FFFFF = 200
  0x00500000 to 0x01FFFFFF = 400
  0x04000000 to 0x3FFFFFFF = 600
  0x40000000 and Above = 800

- Number of Visible Windows (NVW) is scaled as follows (used on the scale of 1000):
  0 – 800
  1 to 3 - 600
  4 to 8 – 400
  8 and above – 200

- Number of Hidden Windows (NHW) is scaled as follows (used on the scale of 1000):
  0 to 10– 200
  11 to 25 - 400
  26 to 50 – 600
  51 and above – 800

4 Training Examples

To apply learning methods of any machine learning concept, a few sets of training examples are required to train/learn. To learn an ANN training example, plays an important role. These training examples set have both positive (self) and negative (non-self) examples as shown in Table 2. In the training examples set of Table 2 there are 20 training examples in which13 are positive (self) and 7 are negative (non-self). Through these examples, training is provided to ANN. The values of 5 processes parameters are used as per the above section 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 will be used instead of actual values as shown in Table 3. This training examples set is constructed, after various running conditions on different workloads of a computer system. The system was virus infected during these observations. Different 20 processes are identified as training set as shown in Table 2. In these 20 training examples thirteen examples are positive examples (generated by system and some application processes) and seven examples are negative examples (generated by viruses).

5 Neural Network Learning

Artificial Neural Network Learning gives a truthful result to approximating real-valued, discrete-valued, and vector-valued target function. An elementary unit to construct ANN is perceptron [17,18].

	PID	FS	MPU	PF	PFPU	PR	FMD	BA	NVW	NHW	Self
P1	136	866584	77236	132687	69836	BN	7/23/2013	0x00400000	0	0	Yes
P2	4060	487424	7956	2340	9088	N	11/24/2008	0x00400000	0	0	Yes
P3	2068	15360	5848	1566	3240	N	4/14/2008	0x00400000	0	5	Yes
P4	2132	3739678	34664	22390	30648	N	1/1/2007	0x00400000	1	10	No
P5	2368	3462552	20020	6304	16756	N	2/1/2012	0x00400000	1	7	Yes
P6	2500	108544	81872	36169	89572	N	12/13/2012	0x00400000	1	15	No
P7	1436	1159168	16636	4811	13400	N	6/10/2012	0x00400000	1	6	Yes
P8	5780	108544	83268	56088	104952	N	10/13/2011	0x00400000	1	12	Yes
P9	3724	263600	6296	1612	3396	N	5/25/2010	0x00400000	0	3	No
P10	3848	221184	12844	3319	9648	N	7/31/2010	0x00400000	0	13	Yes
P11	744	2647432	58528	34487	48700	N	12/6/2008	0x00400000	0	8	Yes
P12	2888	79120	149544	110529	120180	N	11/14/2008	0x00400000	1	12	Yes
P13	2716	15752	8344	8214	6308	N	12/6/2008	0x00400000	0	0	No
P14	3484	16851968	34428	12895	39940	N	9/9/2008	0x00400000	0	55	Yes
P15	3932	1695232	18436	5764	14860	N	4/14/2008	0x01000000	1	17	Yes
P16	1416	1274832	26316	11497	16228	N	11/3/2012	0x00400000	1	11	No
P17	2672	36352	18276	44086	13552	N	5/22/2008	0x00400000	2	12	Yes
P18	664	14336	5256	1342	3156	N	4/14/2008	0x01000000	0	0	Yes
P19	3344	1033728	39104	154032	49800	AN	4/14/2008	0x01000000	4	56	No
P20	1632	911824	21240	5535	12640	N	11/3/2012	0x00400000	0	9	No

Table 2: Training Example Set (in parameter's real value)

A perceptron takes the input as a vector of real-valued, calculate a linear combination of these inputs with their corresponding weights, then produces a outputs 1 if the result is larger than some threshold and -1 if the result is larger than some threshold. Figure 6 shows the Artificial Neural Network to identify the self and non-self operating system process. In this ANN there are ten inputs, five inputs as according to five processes parameters whose parameter values are abbreviated as VL, L, M, H, VH these parameters are Process ID(shown by PID in ANN), File Size(shown by FS in ANN), Memory Peak Usage(shown by MPU in ANN), Page Fault(shown by PF in ANN), are Page File Peak Usage(shown by PFPU in ANN).

Learning an ANN perceptron involves choosing and updating the values of the weights. To learn the perceptron begin with some random weights or with a fix weight value to each weight i.e. 0.5 to each weight and then iteratively apply the perceptron to each example of training data, modifying the perceptron weights as according to the learning algorithm. This process is repeated as many times as needed on the training examples until the perceptron correctly classifies all training examples. Weights are modified at each iteration according to the perceptron training rule by using the learning algorithm. Backpropagation algorithm [7,13,14,15] is a well known learning algorithm for preceptron learning as mentioned below-

	PID	**FS**	**MPU**	**PF**	**PFPU**	**PR**	**FMD**	**BA**	**NVW**	**NHW**	**Self**
P1	VL	VH	M	VH	VH	250	392	200	800	250	Yes
P2	H	H	VL	L	M	500	2071	200	800	500	Yes
P3	M	VL	VL	VL	L	500	2291	200	800	500	Yes
P4	M	M	M	M	M	500	2754	200	600	500	No
P5	M	M	L	M	M	500	924	200	600	500	Yes
P6	M	VL	L	M	VH	500	612	200	600	500	No
P7	L	M	L	L	M	500	795	200	600	500	Yes
P8	VH	VL	H	VH	VH	500	1032	200	600	500	Yes
P9	H	L	VL	VL	L	500	1530	200	800	500	No
P10	H	VL	L	L	M	500	1465	200	800	500	Yes
P11	L	M	M	H	H	500	2059	200	800	500	Yes
P12	M	VL	VH	VH	VH	500	2081	200	600	500	Yes
P13	M	VL	VL	L	M	500	2059	200	800	500	No
P14	H	VH	M	M	H	500	2146	200	800	500	Yes
P15	H	M	L	M	M	500	2291	400	600	500	Yes
P16	L	M	M	M	M	500	652	200	600	500	No
P17	M	VL	L	VH	M	500	2253	200	600	500	Yes
P18	VL	VL	VL	VL	L	500	2291	400	800	500	Yes
P19	H	M	VL	L	H	750	2291	400	400	750	No
P20	L	M	L	L	M	500	652	200	800	500	No

Table 3: Training Example Set (in parameter’s encoding form)

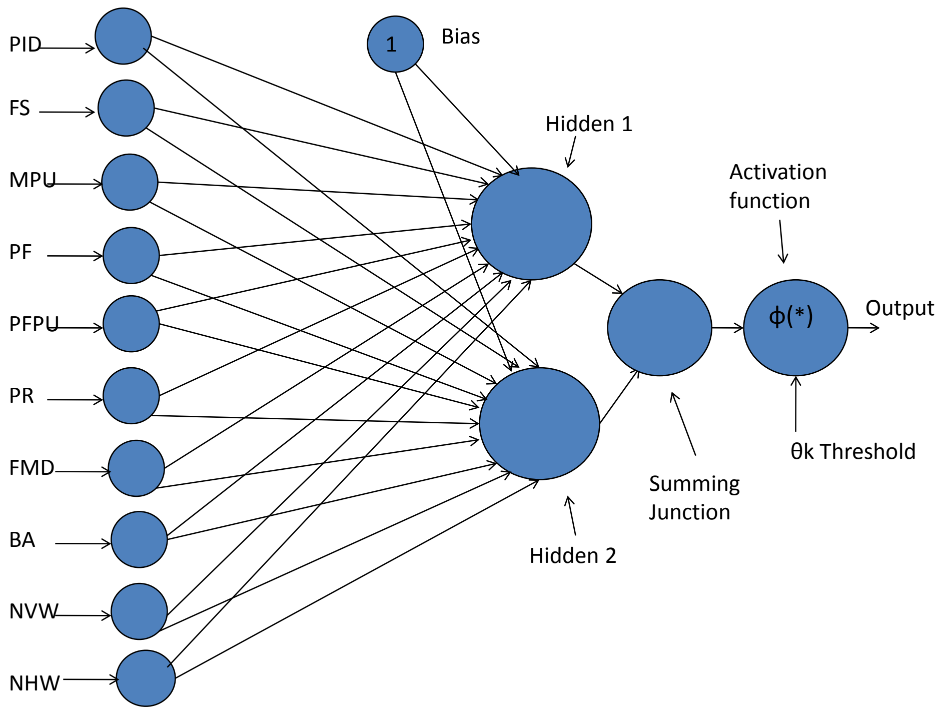


Figure 3: Neural Network for identification of Self and Non-self processes.

5.1 Backpropagation algorithm

Following is the Backpropagation algorithm used in Artifical Neural Network learning - BACKPROPAGATION(*training_example*, η, n_{in}, n_{out}, n_{hidden})

Every training example is a pair of the form ‹ $\vec{x}$, $\vec{t}$ ›, where $\vec{x}$ is the vector of network input values, and $\vec{t}$ is the vector of target network output values.

η is the learning rate (e.g. 0.5), n_{in} is the number of network inputs, n_{hidden} the number of units in the hidden layer, and n_{out} is the number of output values.

The input fiom unit *i* into unit *j* is denoted x_{ji}, and the weight from unit *i* to unit *j* is denoted w_{ji}.

- Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units and n_{iout} outputs units.
- Initialize all network weights to small random numbers (e.g. between – 0.05 and 0.05).
- Until the termination state is met, Do
    - For each ‹ $\vec{x}$, $\vec{t}$ › in training-examples, Do
      Propagate the input forward through the network:

1. Input the instance $\vec{x}$ to the network and compute the output o_u of every unit *u* in the network.

Propagate the error backword through the network:

2. For each network output unit *k*, calculate its error term δ_k
   $\delta_k \longleftarrow o_k\,(1 - o_k)(t_k - o_k)$
3. For each hidden unit *h*, calculate its error term δ_h
   $\delta_h \longleftarrow o_h\,(1 - o_h) \sum_{k \in outputs} w_{kh}\,\delta_k$
4. Update each network weight w_{ji}
   $w_{ji} \longleftarrow w_{ji} + \Delta w_{ji}$
   where
   $$\Delta w_{ji} = \eta\, \delta_j\, x_{ji}$$

The job of the Backpropagation algorithm is to moderate the degrees to which weights are altered at each step. Training examples provide target values t_k only for network outputs, no target values are directly available to indicate the error of hidden units' values. Instead, the error term for hidden unit *h* is calculated by summing the error terms δ_k for each output unit influenced by *h*, weighting each of the δ_k *'s* by w_{kh}, the weight from hidden unit h to output unit k. This weight characterizes the degree to which hidden unit *h* is "responsible for" the error in output unit k.

The Backpropagation algorithm update weights incrementally to obtain the true gradient of error one would sum the *δj xji* values over all training examples before altering weight values. The weight-update loop in the Backpropagation algorithm may be iterated thousands of times in a typical application.

A variety of termination conditions[7] can be used to halt the procedure. One may choose to halt after a fixed number of iterations through the loop, or once the error on the training examples falls below some threshold, or once the error on a separate validation set of examples meets some criterion. The choice of termination criterion is an important one, because too few iterations can fail to reduce error sufficiently, and too many can lead to over-fitting the training data.

5.2 Calculation performed in the ANN

Figure 3 shows the architecture of the ANN used for identification of non-self process. This ANN architecture has an input layer with 10 nodes as according to the 10 process parameters. This ANN architecture has one hidden layer with two nodes, one summing junction to sum the output of hidden layers nodes. Then activation function with threshold function produces the calculated output. Various activation function and threshold function can be used to get better results. In hidden layer nodes following calculation is performed -

$$\sum_{i,\, j=0 \text{ to } n} x_i w_{ji}$$

Where x_i is represents the i'th input node and w_{ji} is the weight of the link of *j* hidden layer to *i* input node. A bias node is added to perform x_0 and w_{01} and w_{20} terms. Same calculation is performed in the summing junction node and scale down on the scale of 1. With the help of activation function sigmodial and threshold value (0.5) this ANN architecture produces the output 0 or 1. If the out put is below 0 then the process is self and if the output is 1 then the process is non-self.

To perform the simple and easy calculation in the hidden layer of ANN, the values of process's parameters have been taken as the mean of the range during the execution of Backpropagation algorithm has been as shown in Table 4.

	Process ID		**File Size**		**Memory Peak Usage**		**Page Faults**		**Page File Peak Usage**	
	Range	**Value used in ANN**	**Range**	**Value used in ANN**	**Range**	**Value used in ANN**	**Range**	**Value used in ANN**	**Range**	**Value used in ANN**
Very Low	714 and below	357	261419 and below	130710	10490 and below	5245	2274 and below	1137	2367 and below	1184
Low	715 to 1938	1327	261420 to 314688	288054	10491 to 31302	20897	2275 to 5358	3817	2368 to 5008	3688
Medium	1939 to 3163	2554	314689 to 4375625	2345157	31303 to 78172	54737	5359 to 25001	15180	5009 to 31269	18139
High	3164 to 4388	3776	4375626 to 8436562	6406139	78173 to 109391	93782	25002 to 43750	34376	31270 to 57530	44400
Very High	4389 and above	7194	8436563 and above	9218281	109392 and above	554696	43751 and above	71875	57531 and above	78765

Table 4: Process's Parameter values during the execution of Backpropagation algorithm (ANN)

From the training example in Table 3, the values of process's parameters 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 will be used 34376. The Backpropagation algorithm has been applied to the training data of Table 3.

The values of process parameters are scaled down according to the desired output. After several of iteration of Backpropagation every weights are set 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. When there will be no update in weights, then the ANN will be fully learned.

After updating the weights of the (training) ANN, this trained ANN is tested with a test data as shown in Table 4. After providing training by training data of Table 3 (after 50 iterations) and then tested by the test data of Table 5, it has been observed that the ANN classify P24, P26, and P29 incorrectly. When the number of iteration increases near about 250 then ANN classify all test data correctly.

	PID	FS	MPU	PF	PFPU	PR	FMD	BA	NVW	NHW	Self
P21	H	M	M	M	M	500	2818.0	600	600	0.2	Yes
P22	M	M	L	M	M	500	691.0	200	800	0.2	No
P23	M	VH	M	M	M	500	2818.0	600	400	0.2	Yes
P24	M	M	M	M	M	500	1967.0	200	600	0.6	Yes
P25	M	VH	M	M	H	500	2818.0	600	600	0.2	Yes
P26	H	VH	L	M	M	500	2818.0	600	600	0.2	No
P27	H	M	L	L	M	500	1689.0	400	600	0.2	Yes
P28	H	M	VL	L	M	500	2291.0	400	600	0.2	No
P29	VH	VH	M	M	H	500	2818.0	600	600	0.2	Yes
P30	M	M	VL	VL	L	500	2291.0	800	600	0.2	Yes

Table 5: Test Data Set

6 Experimental Results and Comparisons

As the number of the iteration of the Backpropagation algorithm increases and weights are updated errors in observing output and desire output decreases as shown in the Figure 4 and Figure 5. Weight update of all input to hidden layer and hidden to output layer are updates slightly in the direction of the positive or negative side as the number of iteration increases as shown in Figure 6. So, to decide when to stop the iteration of Backpropagation algorithm, i.e. how many iteration is sufficient to provide a decision on self and non-self processes. It has been observed from the graph of Figure 4, Figure 5 and Figure 6, that 400 to 500 iterations are sufficient in this case.

It has been observed from the Figure 7, as the numbers of input nodes (parameters) in the ANN increased the security also increased. So, we have to decide that how many input nodes are sufficient. It is cleared from the graph of Figure 7 that when the nodes increases after fifteen then the security remain nearly about constant. It has been observed from the Figure 8, as the numbers of nodes (parameters) in the ANN increases,

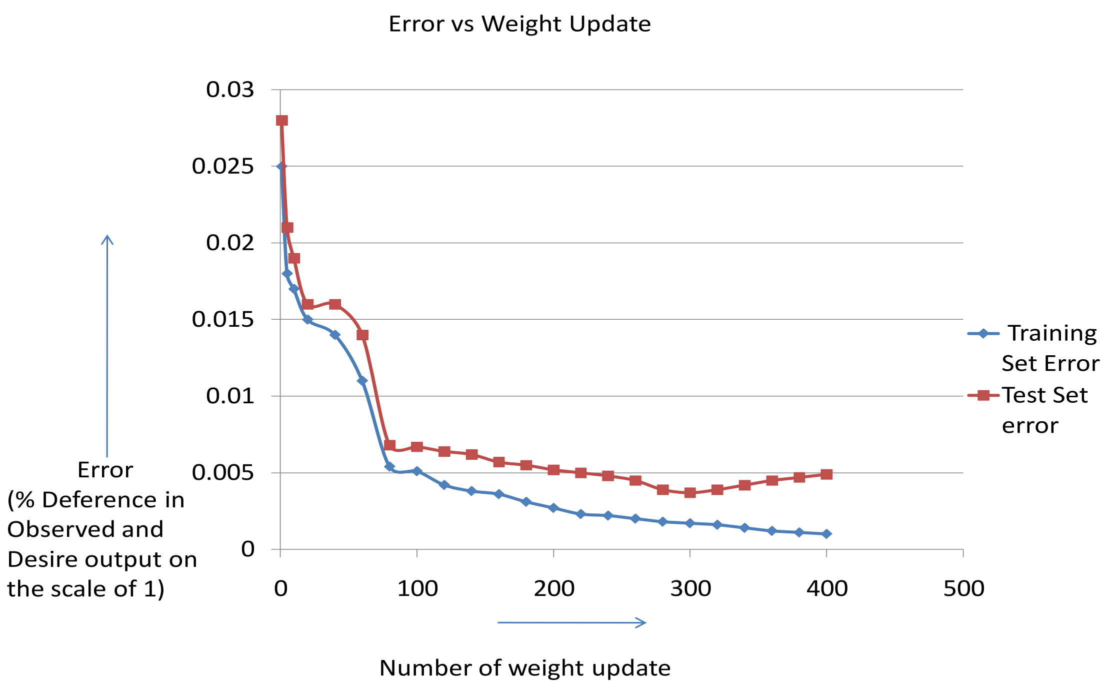


Figure 4: Plots of error E as a function of the number of weight updates

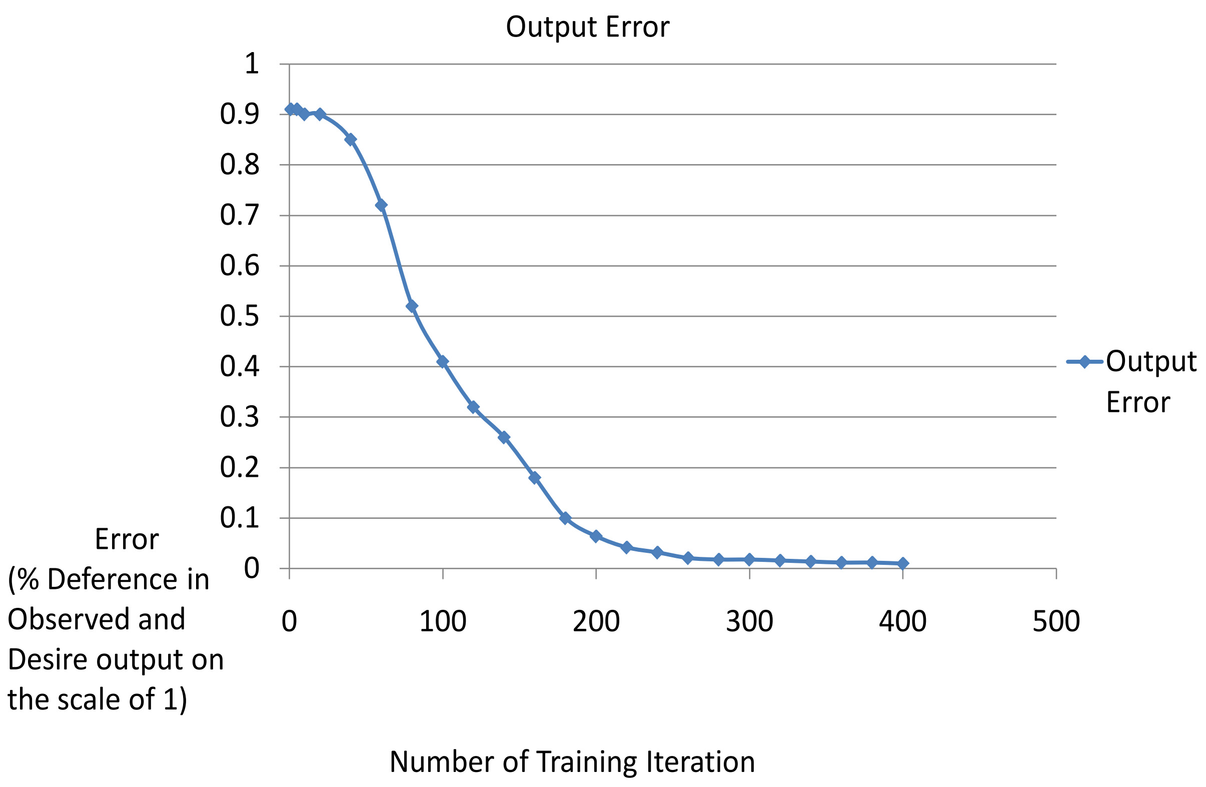


Figure 5: Plots of output error as a the number of iteration increases

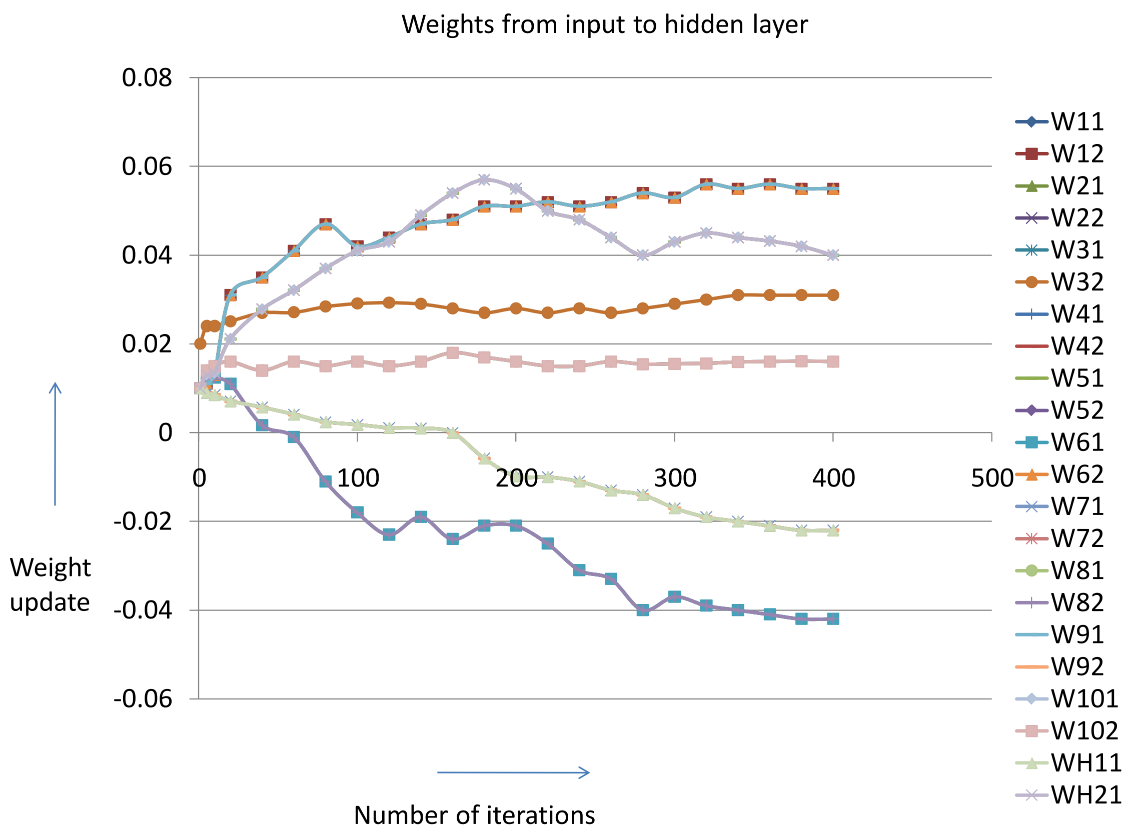


Figure 6: Plots of weight update

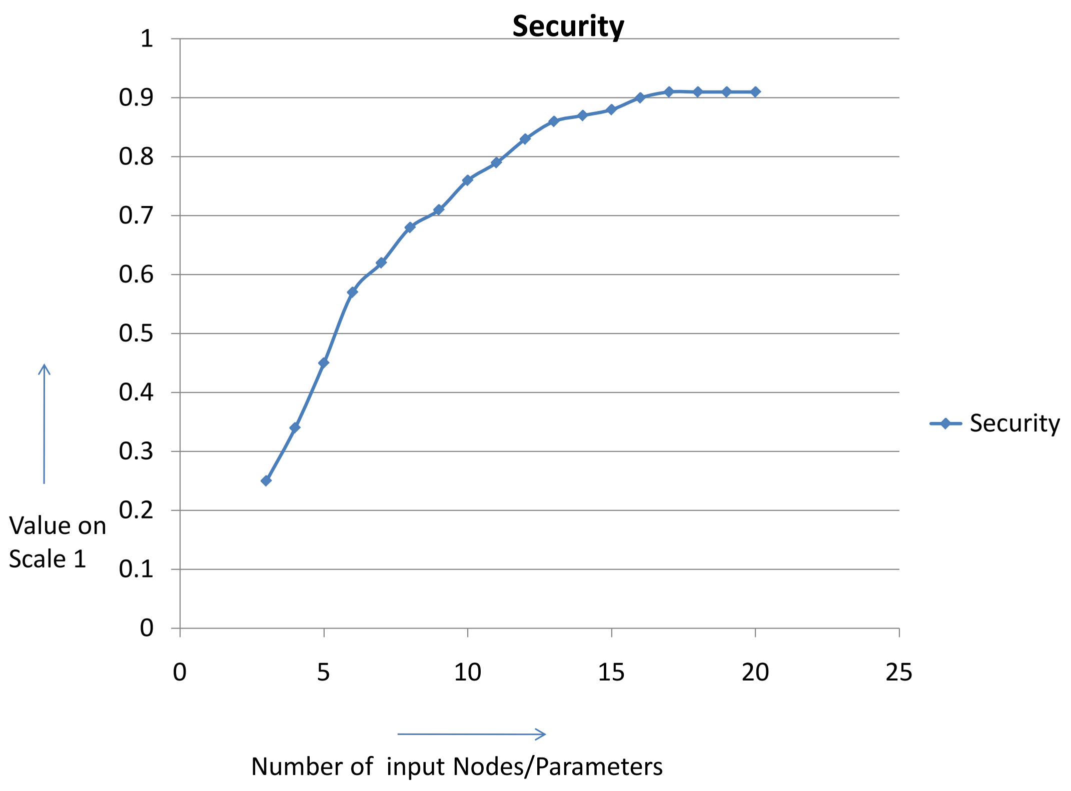


Figure 7: Graph between the number of input nodes and security level

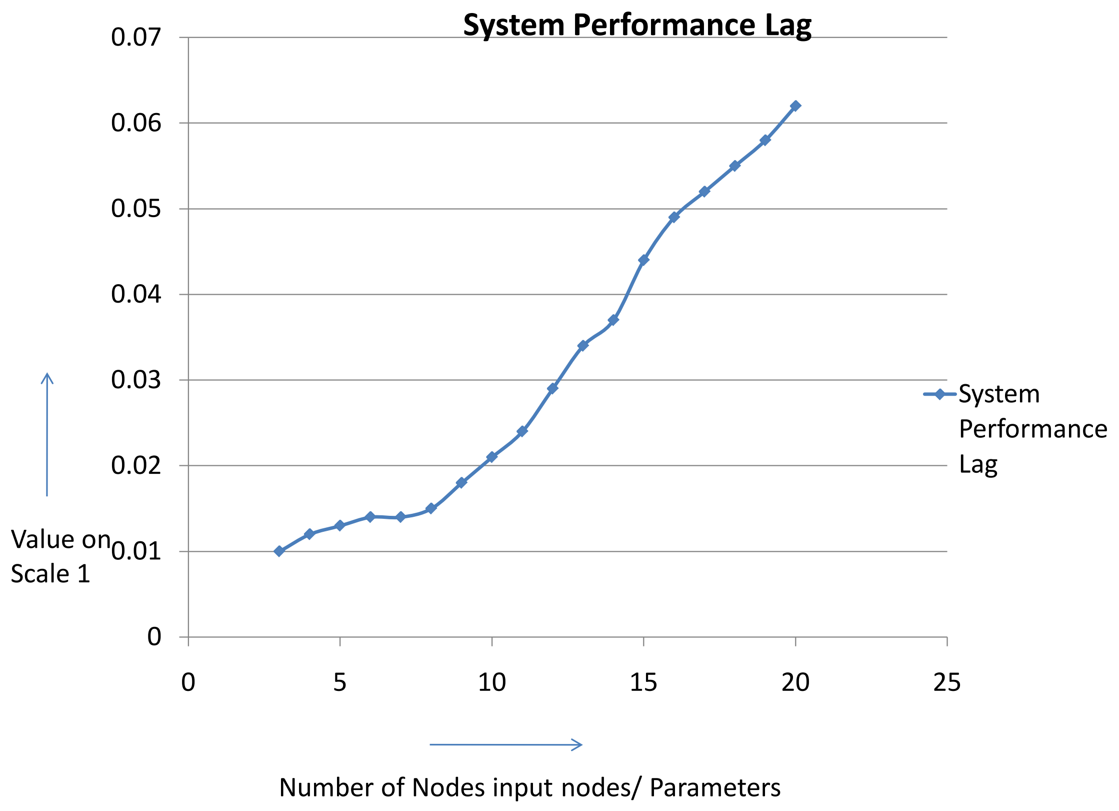


Figure 8: Graph between number of input nodes and decrease in system performance

Parameters / AntiVirus	Scan Time	Accuracy	Detection Rate	Performance Lag	Signature based Detection	Regular Updating Required
AVG Anti-irus	High	High	Normal	Yes	Yes	Yes
Norton Antivirus	High	High	High	Yes	Yes	Yes
Avast Antivirus	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

Table 5: Comparison of Proposed Approach with Some Antivirus tools

there is a performance degradation in the computer system as the processes is checked by the ANN. It is cleared from the graph of Figure 8 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 result plays a very important role.

The proposed approach for security is better than the existing security approach as shown in table 5. The proposed approach takes very less time to find out the non-self processes of viruses as other approaches, scan all the files and data according to the signature (pattern). Accuracy and detection rate of the non-self processes are very high in comparison to existing anti-virus tools, proposed approach check only the processes so, its accuracy and detection rate is high.

As the proposed approach scans all processes so the system performance will become slow. The proposed approach is free from signature updating, as required in existing approaches. As the 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, we can reset the weights and then train with the new training data and test with new test data for updating of the proposed approach.

7 Conclusion and Future work

Various approaches and anti-virus tools are used to provide computer security, but a question arises that these approaches are sufficient or not to provide best security. Artificial Neural Network and other machine learning approaches can play an important role to provide better security to the computer system.

The Artificial Neural Network approach used in this paper to provide better result over the current security approaches. By adjusting the value of learning rate η used in the Backpropagation algorithm once can get the better learn system for identification of non-self processes. By using many bigger training and test data sets, better weight update can be done and it will provide a better result.

References

[1] Rossouw von Solms and Johan Van Niekerk, 2013, "From information security to cyber security", Elsevier's Computer & Security, Vol. 38, pp 97–102.

[2] Cui-Qing Yang, 2003, "Operating System Security and Secure Operating Systems", GSEC-Version1.4, Global Information Assurance Certification Paper. http://www.giac.org/paper/gsec/2776/operating-system-security-secure-operating-systems/104723

[3] http://www.cyberwarzone.com/massive-cyber-security-tools-list-2013

[4] J. K. Percus, O. E. Percus and A. S. Perelson, 1992, "Probability of Self-Nonself discrimination" Proceedings of the NATO Advanced Research Workshop on Theoretical Immunology, Vol. 66, pp 63-70.

[5] S. Forrest, S. A. Hofmeyr, A. B. Somayaji and T. A. Longstaff, 1996, "A sense of self for UNIX processes", Proceedings of IEEE Symposium on Computer Security and Privacy, pp 120-128 http://www.cs.unm.edu/~immsec/publications/ieee-sp-96-unix.pdf

[6] Amit Kumar and Shishir Kumar, 2014, "Artificial Neural Network Based Approach for Identification of Operating System Processes" Journal of Applied Information Science, Vol. 2 Issue 1, pp 1-11.

[7] Tom M. Mitchell, 1997, "Machine Learning", McGraw-Hill International Editions, Computer Science Series, Ch 2 pp 20-50, Ch 3 pp 52-78, Ch 4 pp 81-126.

[8] Haoyong Lv, Hengyao Tang, 2011, "Machine Learning Methods And Their Application Research", IEEE International Symposium on Intelligence Information Processing and Trusted Computing, pp 108 – 110.

[9] Wang Hua, MA Cuiqin, Zhou Lijuan, 2009, "A Brief Review of Machine Learning and its Application", IEEE Information Engineering and Computer Science, ICIECS, pp 1-4.

[10] Derrick H. Nguyen and Bernard Widrow, 1990, "Neural Networks for Self-Learning Control System", IEEE Control System Magazine, pp 18-23.

[11] Guoqiang Peter Zhang, 2000, "Neural Networks for Classification: A Survey",

IEEE Transaction on System, Man and Cybernetics-Part C: Applications and Reviews Vol. 30, No. 4. pp 451-462.

[12] Bart Baesens, Pantelis Bouboulis, and others, 2012, "Neural Networks and Learning Systems Come Together.", IEEE Transactions on Neural Networks, Vol. 23 No. 1, pp 1-6.

[13] Rama Kishore and Taranjit Kaur 2012, "Backpropagation Algorithm: An Artificial Neural Network Approach for Pattern Recognition", International Journal of Scientific & Engineering Research, Vol. 3, Issue 6, pp 1-4.

[14] S. Suresh, S.N. Omkar, and V. Mani 2005, "Parallel Implementation of Back-Propagation Algorithm in Networks of Workstations", IEEE transactions on parallel and distributed systems, Vol. 16, no. 1, pp 24 -34.

[15] S.Jeyaseeli Subavathi and T. Kathirvalava Kumar, 2011, "Adaptive modified backpropagation algorithm based on differential errors" International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.1, No.5, pp 21-34.

[16] CurrProcess v1.13 - Freeware Process Viewer, Copyright (c) 2003 - 2008 Nir Sofer, http://www.nirsoft.net/utils/cprocess.html

[17] James W. Watterson, 1990, "An Optimum Multilayer Perceptron Neural Receiver for Signal Detection", Neural Networks, IEEE Transactions on Vol. 1, Issue 4, pp 280-300.

[18] Sankar K. Pal, Sushmita Mitra, 1992, "Multilayer Perceptron, Fuzzy Sets and Classification", IEEE Transaction on Neural Networks, Vol. 3, No. 5, pp 683-697.