Prediction of Virtual Machine State Based on BP Neural Network

JianPing Li¹, RuiNan Sun¹*

¹ Computer and information technology, Northeast petroleum university, Daqing city, Heilongjiang province, 163318, China
author’s e-mail: 2747893281@qq.com

Abstract. Monitoring software for kernel-level rootkit malware in the cloud computing environment has been relatively perfect. However, these monitors can only ensure the security of virtual machines in the cloud environment, but cannot make a more accurate judgment on the future state of virtual machines. Kernel virtual machine for cloud computing environment safety status of prediction is affected by the impact factors of nonlinear, the traditional forecasting model is based on statistical methods, although dynamic allocation model can be used for the prediction, the ability to overcome the non-linear influence is poorer, predicted results already cannot meet the accuracy requirement of the Internet of things. In order to more accurately predict the cloud computing environment of virtual machine core security status in the future, based on the BP neural network in the virtual machine future state prediction model, and through factor analysis and factor analysis results normalized processing to reduce the size of the input data of BP neural network, the neural and network prediction time. Finally, by processing and simulating the data obtained by libvirt program, the feasibility of the model is verified. It can be seen that the model has a good prediction effect on the future state of the virtual machine.

1. Introduction
Kernel malicious programs of virtual machines often cause great harm to virtual machines in the cloud environment in a short time. Using scientific and effective methods to predict whether virtual machines will be attacked by certain types of rootkit[1] malicious programs in the future is a basis for ensuring the security of the entire cloud computing environment. Therefore, the main research aim of this article is to guarantee the safety of the virtual machine in the next moment, virtual machine future state prediction model is established, using BP neural network[2] network combined with the experiment of index information, prove that the BP neural network algorithm is effective in predicting whether the virtual machine will be attacked by several types of kernel rootkit malware in the future.

2. Analysis of influencing factors
About whether the virtual machine is in safe condition, we adopted four index information of the corresponding index set, in order to facilitate subsequent combined with BP algorithm in anticipation of the future state of the virtual machine, the four indicators of information are: CPU utilization, memory utilization, and network bandwidth utilization and Linux per second the error page number.
Why are the four indicators collected? When CPU utilization, memory utilization and network utilization are low, we consider that the probability of attack is low. Conversely, when CPU utilization, memory utilization, and network utilization are high, the increase in CPU utilization also results in a
significant increase in the number of error pages per second within Linux. The principle behind it is that the attack action taken by rootkit will lead to an increase in CPU load, context switching rate or network utilization, while a high resource utilization means a high probability of attack. Even if their values increase with a normal workload, it is still helpful to prevent attacks from spreading over the running workload when the system is under attack.

CPU utilization is the number one metric that we monitor. CPU utilization, as the name suggests, gives us an idea of how much CPU is being used at a given time. CPU utilization is available by using the `cat /proc/stat` command, which is the most commonly used method of counting CPU utilization in Linux today.

Memory is the second indicator information we need to monitor, because when rootkit is invaded, the utilization of memory will significantly increase, and the memory utilization is too high, other processes that can run normally will not be able to allocate enough memory. We can also check the memory occupation through the top command, but the free command is more commonly used in monitoring:

| Table 1. Linux Memory table |
|-----------------------------|
| Mem | used | free | shared | buffers | cached |
| 40068 | 23205 | 17732 | 487 | 11544 | +/- |
| buffers/cache | 6482 | 24698 |
| Swap | 0 | 0 | 0 |

Let's analyze the contents of the memory record above formally. The memory utilization rate is mainly in the line of Mem. From the figures under the total, we know that the total memory is 40068M. In Linux, memory allocation is similar to the Java virtual machine's memory allocation principle, principle of is lazy, when the system will be a block of memory to memory, when the process after the operation, the system will not immediately put this memory recovery, but over a period of time to observe whether the process will run again, if in this period, the process to run again, the system will not be allocated for this process new memory modules, but use, is the original is ok, it is also a system is a good way to save memory. The above Shared refers to the Shared memory of individual processes, which is used as the occupied part but is generally less used.

We continue to look down, first look at the definition of the swap, it is used to temporarily store memory buffers and cache, in most cases, we should not start them at random, although he would be able to speed up the restart process speed and process operation, but may have some other problems, such as insufficient memory will accelerate swap, speaking, reading and writing, so as to increase the server IO pressure.

Bandwidth utilization is the third type of indicator information that we need to monitor and collect. It is generally used to represent how much a LAN network is used in a certain period of time. It is an important parameter to measure whether the network is crowded, and also the most basic reference factor to measure the network status. If there is no noise and other factors, the bandwidth utilization can be generally calculated by the following formula: bandwidth utilization = total network traffic/theoretical bandwidth X time. We analyze from the Angle of formula, we can know the bandwidth utilization ratio is evaluated over a period of time, and then record the network traffic, of course, this formula is not universal, as just mentioned, it also depends on the evaluation of the different periods, that is to say, to deal with different environments of bandwidth utilization time selection can have significant impact, analysis of sudden flow, the shorter the time, the better, analyze traffic trends, time should be extended. The utilization ratio of the belt is block is inversely related to the network usage. The higher the utilization ratio of the belt be block, the less ideal the situation is. We can also use the bandwidth utilization to determine whether the network abnormal, bandwidth utilization can usually indicate that the network has a packet loss, congestion and other abnormal.

The number of pages missing per second in Linux is the fourth indicator information we need to obtain, because when rootkit invasions the kernel, CPU will generate other exceptions in addition to increased utilization, the most important is that it will cause the system to generate page missing,
which will cause the system to open a separate thread to handle the exception. So, how does the system deal with missing pages? To handle page faults, Linux relies heavily on the page fault exception handler. The program will then look for do_page_fault() to help deal with it. It will set specific methods or steps according to the existing comprehensive environment of the system and specific information of missing pages to solve the problem of missing pages. The do_page_fault() function is powerful because it makes comprehensive use of all kinds of information in the system environment to deal with page fault. Page fault is usually caused by two kinds of exceptions: That is triggered by the kernel page fault anomalies, also known as the kernel mode is unusual, and how to deal with abnormal kernel mode, there is no solution for our reference, it is not in the scope of this article, we just get the purpose of index information, usually by SAR - B in Linux command every second in our we can obtain the kernel of missing pages, i.e., the SAR - B, m, n, m, are taken every few seconds n sampling several times in a row, so, we will be the value of m and n are set to 1, then index information can be obtained.

3. BP neural network based virtual machine state prediction model

3.1 Network construction

3.1.1 Selection of network layers:
(1) simple mapping relationship, in order to speed up the operation speed, in the case of precision requirements can choose the single hidden layer;
(2) complex mapping, although increasing the number of hidden layer can improve the accuracy of network prediction, but this approach will make the network become complex, but also increase the time of data training;
(3) in general, the improved method is to give priority to increase the number of hidden layer neuron nodes.

The prediction of the future state of virtual machine can be regarded as a nonlinear process from the mapping of independent variables (collected continuous partial index information data in the past) to dependent variables (index information in the future period). It can be seen that the future state prediction model of virtual machine based on BP neural network has a simple mapping relationship, so the three-layer neural network structure of single hidden layer can be adopted.

3.1.2 Establishment of prediction model:
Qj(t) (j = 1, 2, 3, ..., m) is the information data at time period j at time t, and m is the total number of time periods. The idea of the prediction model is to use the information data of the first k time periods and this time period to predict the virtual machine state value of the future time period, that is, Q(t), Q(t-1), Q(t-2),... ,Q(t-k) is taken as the input sample, namely the input layer of the prediction model, and Q(t+1) is taken as the output sample. The basic structure of the virtual machine state prediction model based on BP neural network is shown in figure 1
3.2 Network training

The main feature of BP neural network learning process is that samples are transferred from the input layer forward to the hidden layer and the output layer for processing. When there is an error that does not satisfy the accuracy between the predicted result and the given result, the error is transferred backward to adjust the weight between the output layer and the hidden layer and between the hidden layer and the input layer.

Take the index information of the first $P$ periods as the input value, and set the input vector as:

$$X_l=(x_1, x_2, \ldots, x_p) \quad l=1, 2, 3, \ldots, q$$

Where, $P$ is the number of training samples; $Q$ is the number of neurons in the input layer, and the corresponding output vector $Y_l=(y_1, y_2, \ldots, y_s)$, where $S$ is the number of neurons in the output layer. The input values of each unit of the hidden layer are:

$$t_p l \sum_{t=1}^{p} w_{lt} x_p - \theta_t$$

Where:
- $W_{lt}$ is the connection weight between $l$ input layer and $t$ hidden layer;
- $\theta_t$ is the threshold value of hidden layer neurons;
- $R$ is the number of neurons in the hidden layer; Sigmoid function is used as the transfer function. The calculation method of s-type function is shown in the following equation:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Therefore, the output value of the hidden layer is:

$$b_t = \frac{1}{1 + \exp(-w_{lt} x_p + \theta_t)} \quad t=1, 2, 3, \ldots, r$$

By this analogy, the input and output data of the output layer can be obtained as follows:
$L_t = \sum_{f=1}^{r} w_{tf} b_t - \theta_f \quad f=1,2,3, \ldots, v$ (5)

$c_t = \frac{1}{1 + \exp(-\sum_{f=1}^{v} w_{tf} b_t + \theta_f)} \quad f=1,2,3, \ldots, v$ (6)

Where: $W_{tf}$ is the connection weight of hidden layer t to output layer f; $\theta_t$ is the neuron threshold of the output layer. The predicted value of the output layer often has a large error after a training, so it is necessary to adjust the weight of each layer through the back propagation of error. Let the error between the predicted value and the actual value of the training samples in group l be:

$E_t = \frac{1}{2} \sum_{f=1}^{v} (y_f - c_f)^2$ (7)

Given that the global error is:

$E = \frac{1}{2} \sum_{l=1}^{q} \sum_{f=1}^{v} (y_f - c_f)^2$, In order for $E_t$ to get smaller and smaller as a gradient away from it, and $\theta_f$ inversely proportional to its derivative, That is: $\Delta \theta_f = -\frac{\partial E}{\partial \theta_f}$, It can be concluded that the weight adjustment of the output layer is:

$\Delta \theta_f = ad_f b_f \quad f=1,2,3,\ldots,v; t=1,2,3,\ldots,r$ (8)

Where: $a$ is called learning efficiency; $d_f = (y_f - c_f) t (1 - c_f)$, The output layer threshold adjustment is: $\Delta \theta_f = ad_f$. Similarly, the weight adjustment amount from the input layer to the hidden layer and the threshold adjustment amount of the hidden layer can be derived:

$\Delta w_{lh} = \beta e_l x_p \quad l=1,2,3,\ldots,q; t=1,2,3,\ldots,r$ (9)

$\Delta \theta_l = \beta e_l \quad l=1,2,3,\ldots,q; t=1,2,3,\ldots,r$ (10)

Among them, $e_l = \sum_{i=1}^{q} d_f w_{li} b_t (1 - b_t)$. According to the deduced adjustment quantity, the weight and threshold of the hidden layer of the input layer are adjusted.

4. Input and output data processing

4.1 Brief introduction to the experiment process

By controlling the libvirt program[3], we obtain the index information every 1 second. Libvirt is a set of C function library that is free and open source and supports mainstream virtualization tools under Linux. It aims to provide a convenient and reliable programming interface for various virtualization tools including Xen[4], and supports binding with C, C++, Ruby, Python and other mainstream development languages. The default virtualization management tool virt-manager(graphical) and virt-install(command line mode) on current mainstream Linux platforms are all developed based on libvirt. Because kernel-level rootkit malicious programs can be divided into three categories: LKM rootkit, the debug registers hook[5], the inline hooks, we monitored the ordinary virtual machine run three invasion application respectively, and then began to collect data, when an intrusion is LKM rootkit, indicates the result is 1, the invasion of similar, the debug registers invasion of results of 2 hooks, inline hooks the invasion of the results is 3, and the normal state (no violation occurred) monitored results of virtual machines is 0.
4.2 Input and output sample data processing

In the table below, we take the CPU utilization, memory utilization, network bandwidth utilization, and the number of error pages per second as the input factors, and the monitoring results as the output results. Information obtained through libvirt program, 162 indicators, some indicators of information and monitoring results are shown in table 1 below, though the data has been successful, but the input data and output data of the unit is not the same, so you need to normalization processing the data in table 2, can achieve the same unit data, eliminate the influence of adverse factors, the calculation formula of normalized as (11):

\[
Y_i = \frac{X_i - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}
\]  

(11)

In the above formula: \(Y_i\) represents the monitoring result value of the fifth data after normalization, \(X_i\) Represents the original monitoring result value of the ith data, \(X_{\text{max}}\) represents the maximum value of the original monitoring result, In the same way, \(X_{\text{min}}\) represents the minimum value of the original monitoring result, The data results of normalization are shown in table 3.

| CPU utilization | Memory utilization | Bandwidth utilization | Error pages per second | The results |
|-----------------|-------------------|----------------------|------------------------|--------------|
| 0.610893        | 0.010931497       | 0.21972527           | 10582.18               | 0            |
| 0.6070125       | 0.007182298       | 0.23877195           | 48200                  | 0            |
| 0.6325549       | 0.0176118         | 0.2238236            | 6976                   | 3            |
| 0.63058484      | 0.06124722        | 0.24670693           | 27655.45               | 3            |
| 0.6066358       | 0.002501212       | 0.2318217            | 2500                   | 3            |
| 0.66295314      | 0.040974952       | 0.24379402           | 5154                   | 1            |
| 0.60933214      | 0.007876126       | 0.22887155           | 2604.95                | 0            |
| 0.62033015      | 0.010863614       | 0.2294591            | 5547                   | 3            |
| 0.88403934      | 0.005582766       | 0.30311048           | 38966                  | 2            |
| 0.63605845      | 0.021908967       | 0.23970912           | 37873.74               | 3            |
| 0.6055632       | 0.002075887       | 0.2469283            | 1127.72                | 0            |
| 0.8840029       | 0.00418           | 0.3029243            | 34395.92               | 2            |
| 0.62553656      | 0.008544776       | 0.21717232           | 13549                  | 3            |
| 0.6216009       | 0.006355878       | 0.2395943            | 2833                   | 3            |
| 0.66212374      | 0.041679095       | 0.25437713           | 39110                  | 1            |
| 0.6276199       | 0.011540314       | 0.2420076            | 35697                  | 3            |
| ......           | ......             | ......                | ......                  | ......        |
| 0.88409954      | 0.014172251       | 0.29314184           | 5662.38                | 2            |
| 0.61371374      | 0.06867782        | 0.241719             | 35685.86               | 0            |

| CPU utilization | Memory utilization | Bandwidth utilization | Error pages per second | The results |
|-----------------|-------------------|----------------------|------------------------|--------------|
| 0.028259875     | 0.026968544       | 0.035325237          | 0.176935817            | 0            |
| 0.014462733     | 0.016868813       | 0.238145084          | 0.815161101            | 0            |
| 0.105278896     | 0.044964192       | 0.078966578          | 0.115772964            | 1            |
| 0.098274335     | 0.162510902       | 0.322641249          | 0.466611359            | 1            |
| 0.013123374     | 0.00425873        | 0.164134881          | 0.039835129            | 1            |
4.3 Establishment of the network

The prediction model is a three-layer structure model, with four neurons in the input layer and one neuron in the output layer. The number of neurons in the hidden layer should be satisfied: \[ n < \sqrt{a + b + c}\], where: \( n \) is the number of hidden layers, \( a \) is the number of neurons in the input layer, and \( b \) is the number of neurons in the output layer; \( C \) is a constant between 0 and 10, with input layer 4 and output layer 1. If \( c \) is 10, the number of hidden neurons cannot exceed 12. The accuracy of the model increases with the increase of the number of hidden neurons, but at the same time, the network structure tends to be complex, thus increasing the calculation time of prediction results. Under the premise that the satisfied precision of the results, should try to simplify the network results, in order to quickly get and measurement, so use one input layer (It contains four neurons) neurons, one hidden layer neurons (It contains ten neurons), 1 output layer (It contains one neurons) neurons of the BP neural network, through the simulation results can be seen, can not only guarantee the accuracy of the results, and results quickly and will see the result of the experiment analysis section.

4.4 Analysis of experimental results

I selected 110 pieces of data as training samples, and the remaining 52 pieces of data as test samples. I set the number of hidden nodes as 10, the learning rate as 1.5, and the maximum number of iterations as 4500. The prediction was carried out in the BP neural network prediction model after training. The training results are as follows:

Maximum training error:0.52742----Mean training error:----Mean square training error:0.055902-----Training recognition rate:99.0909
Maximum test error:2.9965----Mean test error:----Mean square test error:0.52736----Test identification rate:78.8462

The variation curve of training error of BP neural network algorithm is shown in figure 2 below:
5. Conclusion
The indicator information in this article is mainly based on the libvirt program, such as CPU utilization, form under the premise of index set, combining with BP neural network algorithm to make the future state of the cloud computing environment virtual machine forecasting model, after the experiment proved that the training sample test rate is 78.8462, shows that the prediction effect is good, the BP algorithm is effective in predicting whether the virtual machine kernel will be attacked again by rootkit malicious programs in the future.

References
[1] Yan Han,Yan. Xiao Fei,Zhang.Liu Pei,He. (2016) Analysis of Rootkit hiding technology [J].Journal of Beijing university of electronic science and technology.,24(02):7-11.
[2] Jin Zhou, Xia.Xiang Yang ,Wu .(2019) Prediction of coal seam gas content based on BP neural network[J].The wind of science and technology.,08:185-226.
[3] Wei, He. Feng,Li.(2018) Research on the construction of virtual resource pool based on kvm-qemu and Libvirt [J].Computer products and circulation.,12:130-131.
[4] Yong ,Song .Zhi Ping, Cai.(2018) A construction method of cloud infrastructure management platform based on Xen [J].Journal of changchun normal university.,37(10):21-27.
[5] Wen Chang, Shi.Hong Wei,Zhou.Jin Hui, Yuan.Bin,Liang.(2012)Detection of kernel exception hook using hardware debugging features [J].The Chinese communication,9(10):78-90.