Detection of Viral Activity using Anomaly Detection Techniques over System Profile

Mrityunjay Gautam
Department of Computer Science and Engineering
Indian Institute of Technology
Kanpur,India.
Email : mrityu@cse.iitk.ac.in

***Abstract*— We have witnessed the release of multiple viruses and worms in a few years. Some of these worms are meant for some attack on a particular website, or any application in the system. Whereas, others damage some data on the system. Following any release of a virus or a worm, we find a new patch being released by the vendors, or some virus updates being released by the antivirus companies. This project was aimed at designing the first step of a new kind of antivirus system which does not work like the general commercial antiviruses, but it works like the human immune system. The idea behind such a system is that this system will be able to detect and remove any kind of old or new virus or worm from the computer without putting an extra patch every time. The design of the immune system in a computer is a later step after this project which may include the design of antibody like mechanisms in the system. But before the antibodies can attack any new virus, the system should realize that it is under attack of a virus. In this project, I have worked on the virus detection. The project uses anomaly detection technique for recognizing any malicious state in the system.**

I. Introduction

We have witnessed the release of multiple viruses and worms in a few years. Some of these worms are meant for some attack on a particular website, or any application in the system. Whereas, others damage some data on the system. Following any release of a virus or a worm, we find a new patch being released by the vendors, or some virus updates being released by the antivirus companies.

This project was aimed at designing the first step of a new kind of antivirus system which does not work like the general commercial antiviruses, but it works like the human immune system. The commercial antivirus systems are signature based. They have a database of signature of all known viruses and worms which they try to match against every file which they suspect to have a possible virus. This method is very good at detecting the viruses and worms whose signatures are already known. But they are unable to detect the attack of any new worm or virus. The main reason behind this is that the signature of this new virus is not known. The idea behind developing this kind of a system is that it will be able to detect and remove any kind of old or new virus or worm from the computer without putting an extra patch every time. The design of the immune system in a computer is a later step after this project which may include the design of antibody like mechanisms in the system which will attach itself to the virus and deactivate the viral activity. But before the antibodies can attack any new virus, the system must be capable of realizing that it is under attack of a virus or a worm. In this project, I have worked on the virus detection.

The project uses anomaly detection technique for recognizing any malicious state in the system. This paper will describe in brief, the idea of a computer immune system, anomaly detection systems, and finally the paper will describe the details of the virus detection system that was developed in this project.

II. Computer Immune System

Before explaining how the computer immune system works, or is expected to work, I would briefly describe how the human immune system works. The idea of a computer immune systems is inspired by the immune system which can be observed in any of the higher vertibrates. Whenever the body is exposed to any pathogen which may be a virus, bacteria or any other foreign object, the body responds to it using the immune system. This innate immune system profides the first line of defense, failing which, we need to take extra precautions and cures. But a lot of problems are solved by the immunity itself.

To discriminate the harmful pathogen from the non-harmful cells (essestially all the body cells), the antibodies have some knowledge of the ”self” combined

with some knowledge of the known pathogens. The biological immune system recognizes the self by the help of some specific proteins which are present in the body, but are not generally found in the the pathogens. Similarly the body recognizes the pathogens which it has seen previously. This information is encoded in the RNA of the cells.
The concept of a computer immune system is very similar to the above described biological immune system. The major parts of the computer immune system as described in [1,2] are as follows:

1) Detecting the presence of a virus or a worm on the user's system.
2) Capturing the state of the system and a sample of the virus and sending it to a central administrative computer for analysis.
3) Analyzing the virus sample to find the cure of the virus which includes detection and removal.
4) Delivering and applying the cure to the infected computer.
5) Distributing the prescription to all the computers in the network.

For details of each of these steps, please refer to [1,2]. In this project, only the step 1 of the computer immune system has been targeted. This project uses the idea of "Anomaly Detection" for detecting any viral activity in the computer. In the next section, I will talk about anomaly detection in general.

III. Anomaly Detection

Anomaly detection is a technique of detecting any deviant behaviour of the computer whose behaviour is being monitored by the system. This paper will discuss two different ways of anomaly detection:

1) Profile Based Anomaly Detection
2) Signature Based Anomaly Detection

A. Profile Based Anomaly Detection

A behaviour is called anomalous if it deviates significantly from the normal behaviour. So for anomaly detection, the first thing that needs to be done is to learn the profile of the system. Most precios work done in the area of anomaly detection has used the profiles for user behaviour and uses this profile for matching against the system behaviour. The anomalies are detected using these profiles which are either statistical in nature or they are learnt using some machine learning technique like neural networks.
Another method of building system profile as described in [3,4] is by building a profile of every root level process running on the system and mointor every process for possible anomalies. This paper describes a way to build profile for any privileged process using the sequence of system calls it make. In general, any privileged process makes the same sequence of system calls at any point in time. This method builds a pattern of system calls and tries to find an anomalous pattern.

1) Advantages:

- Possible to detect new viruses and worms because this method is not dependent on any predefined signatures.
- Does not need any update or patches on the advent of a new virus.
- Possible to detect viruses and worms which do not reside on the file system but get implanted in memory only.

2) Disadvantages:

- Difficult to describe a heuristic which will work on all kind of computer systems.
- Probability of false alarms are high, both for false negative and false positive depending on the threshold.

B. Signature Based Anomaly Detection

Another method of anomaly detection is signature based anomaly detection which is used by most of the commercial antiviruses available today. They have a database of signatures of all the known viruses and worms and they try to match these signatures in any file they find doubtful of infection.

1) Advantages:

- Probability of false alarms is extremely low.
- Perfect detection of the known viruses and worms.

2) Disadvantages:

- Detection of new viruses or worms is not possible.
- Can not detect any viral activity which is not present in the file system.

Since the signature based anomaly detection techniques are incapable of detecting new viruses, and the idea of building a computer immune system is to detect new viruses, I have chosen the profile based anomaly detecion technique in this project.

IV. Project details

A. Approach

In this project, I have tried to build an anomaly detection system based on system profile. This system profile does not include the profile of every single process based on their system calls as described in [3,4]. In this project, we have used a complete system profile as a whole. The feature vector which has been used in this project for building the profile can be broadly classified into 6 main parts depending on which part of the system it corresponds to:

1) CPU Parameters
2) Process Parameters
3) Memory Parameters
4) Disk Parameters
5) File System Parameters
6) Network parameters

These set of features are then learnt using an unsupervised learning algorithm. We can not use supervised leanring algorithm because we can not label the data into its corresponding clusters manually. We can use any algorithm which classifies the data in predefined set of clusters because we do not know the number of clusters before hand. So, we need to use some algorithm which makes the profile of the system from the given training sample and marks all the natural clusters. There are many algorithms available for such kind of clustering like, self-organizing maps, agglomerative clustering algorithms and dissociative clustering algorithms. In this project, I have used an agglomerative clustering algorithm whose details will be given in the next section.

B. Implementation

The set of paramters that i have used in this project implementation are as follows. Each of these data is taken from the /proc file system every 1 minute by a java program.

1) CPU Parameters:

1) Average Number of ready to run processes waiting for the CPU since the last 5 mins.
2) Time spent in performing the idle process in last interval.
3) Time spent in performing user level processes in last interval.
4) Time spent in performing kernel level processes in last interval.

2) Process Parameters:

1) Number of runnable processes
2) Number of processes waiting for interrupt
3) Number of zombie processes
4) Number of processes swapped out
5) Number of system level processes actives
6) Number of processes active

3) Memory Parameters:

1) Fraction of pages free
2) Number of pages brought in from swap in last interval
3) Number of pages swapped out in last interval
4) Fraction Memory utilization
5) Fraction Swap Utilization
6) Fraction of pages shared
7) Fraction of page soft page faults in last interval

4) File System Parameters:

1) Change in number of allocated file descriptors over last interval
2) Change in number of free allocated file descriptors over last interval

5) Disk Parameters:

1) Number of page hard page faults in last interval
2) Number of blocks read in last interval

3) Number of blocks written in last interval

6) Network Parameters:

1) Number of packets sent in last interval
2) Number of erroneous packets sent in last interval
3) Number of packets sent which were dropped in last interval
4) Number of packets received in last interval
5) Number of erroneous packets received in last interval
6) Number of packets received which were dropped in last interval

A total of 28 paramters are taken in this project to build a system profile for a linux based computer. The algorithm which is used for clustering is an agglomerative clustering algorithm. Given the training data points, the algorithm starts forming clusters by joining two clusters which are closest to it. The clusters may consist of a single point or it may be a set of multiple points which are connected with each other as a minimum spanning tree. This algorithm goes on joining the clusters untill it gets a complete minimum spanning tree. Now, the algorithm breaks the longest edges which are above a set threshold. So, if a total of **k** edges are broken, we get a forest of minimum spanning trees with **(k+1)** trees.
The above method of building a **MST** is by using the Kruskal's Algorithm as described in [5]. But since we finally need as **MST**, so in the implementation of this project, an alternate algorithm has been implemented called the Prim's Algorithm [5]. This algorithm builds a MST by starting from a single node as a root. Then it joins that node from the rest to the **MST** which can be connected to the **MST** with an edge of minimum weight. This is a greedy strategy of building a **MST**.
The measure for similarity that is used for our project is a weighted euclidean distance of these parameters. The weights consist of two parts: Normalization Factor and Weights. The normalization factor is important because the values that we get for each of the parameters are in different orders of 10. So, I have used a normalization to bring each of these values to the order of 0.1. After that, the weights are the heuristic values that I have used for anomaly detection. Keeping in mind the typical viral activities, the weights assigned to the network traffic and to the disk reading and writing is a bit higher than other weights.
The training algorithm generates a set of clusters after the training on the four days sample. These clusters are assumed to be spherical for simplification of calculations and to reduce the computational time. So, each of these clusters contains a cluster center which is calculated as the mean of all the nodes in the cluster; and a cluster radius. The radius of the cluster is assumed to be the distance between the cluster mean and the farthest point from the cluster center. Both these informations are then dumped in a file called "cluster.txt" which is then read by the Predictor for deciding whether any given point is anomalous or not. The predictor defines a variable called **THRESHOLD** which maintains the deviation allowed from the cluster radius for any given point to be marked as a **NORMAL** point. For the current implementaion, this value has been set to 1.1 which means that the deviation allowed is 10 percent.
The predictor has been implemented in java and it collects the data from the /proc file-system in the same way as the program for collecting the data for training was doing. This sampling of the system is done every five second. This sampled data is then matched against all the known normal clusters. If a match was found with any of the present clusters, then this data point is marked as NORMAl and no action is taken further. But if this data point was found to be off from all the clusters and their threshold then this point is assumed to be anomalous and a messgage is fired on the console telling that this process was anomalous. For later processing of the anomalous, the complete data point and the system snap is stored in a file called **"anamoly.txt"** which can be used by the system administrator for further analysis. This system snapshot consists of the dump of **ps -f -e** which lists all the processes running on the system at that instant and the user who are the owner of these processes.

V. RESULTS

The training data that is collected has been collected at an interval of one minute for four days on a computer in the security lab, IITK. This data was used to build the profile for monitoring the use of the computer. While the data collection process was in progress, some sftp connections were made and some large files were downloaded from the machine. This allowed for the machine to collect data when the machine was undergoing some network transactions.
In the prediction phase of the application, the data is being monitored every 5 secs. So the relevant

parameters, which need to be scaled up to 1 min value has been multiplied by a factor of 10.0. The anomaly detection system has been tested on many kind of possible anomalous behaviour. It was not possible for us to create an actual anomaly by spreading a virus or any worm on the linux system and try to detect it. So, only a limited data of four days was taken, and some kind of processes that were not present in those four days were tried for anomaly detection. The results were as follows:

- The training algorithm generated a total of 26 clusters using 5820 data points.
- The system was able to detect the condition of excessive disk writes as an anomaly.
- Any kind of large file upload was marked as anomaly assuming the file upload was being done via sftp
- Uploads of multiple small files was not marked as anomalous
- Download of either large or small files was not anomalous. This can be explined by the existence of the sftp download data in the training smaple.
- If there was a ssh connection with the machine and the user is running some heavy application, then this state is still a normal state. The heavy applications that were tested were kate and mplayer.
- The system generates one kind of false alarm. This alarm is generated when any process which has done a lot of page faults, eg a sftp download process, is terminated, the system marks this behaviour as anomalous. This is being counted by the system as negative page faults because these many pages are now freed after the process is killed. May be this needs some redefinition of the hard page fault parameter or possibly, it needs addition of some more parameters. How exactly this false alarm can be removed is still not very clear.

VI. Possible Future Work

This project has a lot of scope for additions and research. Some of the possible extensions to this project can be:

1) One obvious extension of this project is the design and development the computer immune system
2) Currently, I have not been able to go through extensive testing of this anamoly detection system using some real life viruses and worms. A possible extension to this work will be to test the system with some real viruses and come up with a good heuristic function to define the similarity measure between points in the training data. One of the possible methods can be to use some optimization algorithms like genetic algorithm to come up with a good heuristic function.
3) The values of **THRESHOLD** is currently being set manually. A extension to this work can be to actually learn a good value of the **THRESHOLD** so that the best natural clusters can be formed.
4) This current feature vector consists of 28 features. It might be possible to capture a better profile of the system by adding some modules to the existing /proc file system so that some more system related information can be extracted. The change that needs to be done in the existing program are as follows - redefintion of the similarity function, resetting the paramter values accordingly and possibly adding a few more parameters. Rest complete program remains untouched. Hence the main work can be writing the modules in the proc file system.

VII. Acknowledgements

I am grateful to Dr. Dheeraj Sanghi and Dr. Deepak Gupta for the extreme support and guidance I got from them during this project. This project would not have been a success had it not been for them. I express my thanks to both my instructors who helped me throughout the semester so that I can complete this project. I am also thankful to other members of security group who helped me in testing and training phases of my project.

References

[1] Jeffrey O. Kephart,Gregory B. Sorkin, Morton Swimmer and Steve R. White, IBM Thomas J. Watson Research Center. *Blueprint for a Computer Immune System*, Virus Bulletin International Conference, October 1-3, 1997.

[2] Steve R. White, IBM Thomas J. Watson Research Center. *Open Problems in Computer Virus Research*, Virus Bulletin International Conference, October 1998.

[3] Stephanie Forrest, Steven A. Hofmeyer, Anil Somayaji and Thomas A. Longstaff. *A Sense of Self for Unix Processes*, IEEE Symposium on Security and Privacy 1996 IEEE Computer Society Press, Los Alamitos, CA pg 120-128

[4] Patrik D'haeseleer, Stephanie Forrest and Paul Helman. *An Immunological Approach to Change Detection: Algorithms, Analysis and Implications*, IEEE Symposium on Security and Privacy 1996 IEEE Computer Society Press, Los Alamitos, CA pg 110-119

[5] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest. *Introduction to Algorithms*, Prentice-Hall of India Private Limited, New Delhi, India. Fourth Indian Reprint, May 2001.