Using Data Analysis by deploying Artificial Neural Networks to increase honeypot security

Milad Daliran
Department of Computer Engineering
Islamic Azad University
Arak Branch, Iran
engineer.daliran@gmail.com

Ramin Nassiri
Department of Computer Engineering
Islamic Azad University, Central Tehran
Tehran, Iran
r_nasiri@iauctb.ac.ir

GolamReza Latif- Shabgahi
Control and Computer Dept,
University of Power and Water Industry
Tehran, Iran
g.latif@pwut.ac.ir

***Abstract*- The goal of this research is to increase honeypot security through data analysis with Artificial Neural Network (ANN). Thus, first we present an approach to detection presence of computer malcode in the honeypot based on ANN while using the computer's behavioral measures. Then, we identify significant features, which describe the activity of a malcode within a honeypot, by acquiring these from security experts. We suggest employing fisher's score, one of the feature selection techniques, for the dimensionality reduction and identification of the most prominent features to capture efficiently the computer behavior in content of malcode activity. Later on, we preprocess the dataset according to this technique and train the ANN model with preprocessed data. Finally, we evaluate the ability of the model to detect the presence of a malcode in the honeypot when honeypot is at risk.**

I. INTRODUCTION

Increasing security of honeypot has been substantially subject of research during the past years [1][6]. The Honeypots can be considered as powerful tools which make it feasible to collect extensive information on a variety of threats. To obtain this information, one has to allow attackers and malicious code to have potentially privileged access- to honey pot. As a result, the price paid for this capability is *risk* [10]. The term "malicious code" (malcode) commonly refers to pieces of code, not necessarily an executable file, which are intended to harm, whether generally or in particular, the specific owner of the host. One type of malcode is Virus, which injects its code into an innocent file (a host) and is executed whenever the file is executed. Viruses are not required to gain user intervention to propagate. Other types of malcode are Worms, which actively propagate, exploiting vulnerability in the operating system or in a program installed through communication protocols. Other malicious codes include Trojans, which are computer programs that have a useful functionality, but also have some hidden malicious goal, and backdoors, which enable remote access and control with the aim of gaining full or partial access to the infected system. A recent survey of intrusion detection [9] suggests using artificial intelligence (AI) techniques to recognize malicious code in single computers and in computer networks [2]. One of the AI techniques is Artificial Neural Network (ANN) [7], which we deploy it to increase honeypot security. The main advantages of ANN employment are high level of accuracy in real-time operation, low CPU resources utilization during the classification phase and the ability to generalize, in order to detect and identify, any previously unseen classes [5]. In this paper a different approach based on AI techniques is suggested for increasing honeypot security against malcodes. The detection of the presence of a malcode [4] in a honeypot is performed by analyzing the overall computer behavior. We define the computer behavior by a variety of different values which can be measured in the computer while is operating. In this study we select a few number of measured features by using fisher's score technique, while increasing the detection accuracy and decrease overall computational, so that the attacker will not understand how to capture system behaviors.

II. HONEYNET AND RISK

A. HONEYNET

Honeynet is a high-interaction Honeypot designed to capture in-depth information. It is a highly controlled network where every packet entering or leaving is monitored, captured and analyzed. There are two key requirements that a Honeynet must implement: data control and data capture. Data control defines how activity is contained with the Honeynet without an attacker being informed of it. Its purpose is to minimize risk. Data capture is capturing all of the attacker's activity without providing information for the attacker. Information has different organization. Also, organization have different threshold for risk.

B. Risks and the reduction of their destructive effects

With reference to risk, there are four general areas we have to cover: Harm, Detection, Disabling and Violation [10]. We present approaches to reduction of their destructive effects.

1) Harm

Harm is when honeynet is used to attack or harm other non-honeynet systems. Data control is the primary means of mitigating this risk. However, there is no guaranteed method to ensure that a honeynet can not be used to attack or harm someone

...

else. No matter that mechanisms are put in place an attacker can eventually bypass them. Against this risk when honeynet is recognized as intruded the honeynet can be disconnected from network and necessary actions to avoid the risk may be performed by network administrator.

2) Detection

Once identifying a honeynet, its value is dramatically reduced. Attacker can ignore or bogus the honeynet, eliminating its capability for capturing information and even more dangerous, an attacker can produce false information into a honeynet, misleading your data analysis. When honeynet is detected by the intruder, false data may be produced, so if we recognize the risk the data are completely wrong and shouldn't be referred anymore.

3) Disabling honeynet functionality

This could be an attack to disable either data control or data capture capability, potentially without the honeynet administrator knowing that functionality has been disabled. The attacker could feed the honeynet with bogus activity, making administrator think data capture is still functioning and recording activity, While is not. When data control or data capture are disabled by the intruder and the honeynet is occupied false activities to distract the network administrator, by analyzing system behaviors the presence of the malicious codes is understood to perform avoiding actions.

III. CLASSIFICATION, FEATURE SELECTION METHOD

A. classification method

We used a typical feed forward neural network, together with the levenberg-marquardt training method [8]. The number of hidden neurons was set to seven.

B. Feature selection technique

Thus continuous monitoring of large number of measured features may demand a significant part of the computational resources which may attract the attention of the attacker about background data capture then the recorded data may be changed by the attacker. We use a features selection method to reduce number of measured features. This technique is Fisher's score ranking, which calculates the difference, described in terms of mean and standard deviation, between the positive and negative examples relative to a certain feature. Equation (1) defines the Fisher score, in which *Ri* is the rank of feature *i,* describing the proportion of the substitution of the mean of the feature *i* values in the positive examples (*p*) and the negative examples (*n*), and the sum of the standard deviation. The bigger the *Ri*, the bigger the difference between the values of positive and negative examples relative to feature *i;* thus, this feature is more *important* for separating the positive and negative examples [5].

$$R_i = \frac{|\mu_{i,p} - \mu_{i,n}|}{|\sigma_{i,p} - \sigma_{i,n}|} \quad (1)$$

IV. DATASET DESCRIPTION

Since no public standard data set was available for this work, we had to create our own dataset. It contained six real malcode. The number of features we measured was 160. We used windows performance tool, which enables monitoring of system features that appear in these main categories: Internet Control Message Protocol (ICMP), Internet Protocol (IP), logical disk, memory, network interface, objects, paging file, physical disk, process, processor, system , TCP, thread, UDP. The windows performance tool was configured to measure the features every second for 20 minutes for every malcodes. In addition there was a time period in which no malcode was activated. In the rest of the paper we will refer to this clean period as a state named "clean". The resulting dataset were combined into a single dataset. We describe the specification of dataset, the environment in which it was generated and malcodes we used.

In order to perform the evaluation in realistic environment, we considered to major aspect: computer hardware configuration and user activity.

Computer hardware configuration: computer ran MS windows XP. Using a pc based on Pentium IV 3.5 GHZ, bus speed 800 MHZ and memory 1GB.

User activity: a user opened several applications, including MS-word, Excel, Internet Explorer and windows Media player in a scheduler order. The exact schedule of the user activity is shown in Table I.

During the evaluation we used two Trojans, two worms and two viruses. Each one was injected separately into a clean computer for a constant length of time. While selecting worms from the wild, our goal was to choose malcodes that differ in their behavior, from among the available malcodes. Some of the malcodes have a heavy payload of Trojans to install in parallel to the distribution process upon the network; others focus only on distribution. Another aspect is that they have different strategies for IP scanning which results in varying communication behavior, CPU consumption, and network usage.

TABLE I
USER ACTIVITY SCHEDULE

Time period	*User activity*
0-5 minutes	*Opens 10 MS-word files* *Downloading three files simultaneously*
5-10 minutes	*Opening 5 MS-Excel files* *Generating random numbers in MS excel* *Two files downloading* *Listening to internet radio*
10-15 minutes	*Open 13 MS excel files* *Generating random numbers in MS excel* *Browsing the internet with using internet explorer*

...

While all the malcodes are different in nature, so we have to find common characteristics Thus We briefly describe here the main characteristics, relevant to this study, of each malcode included herein. We briefly describe the six malcodes we used:

1) Delf.AAM

Delf.AAM is a Trojan, which allows getting into the affected computer. It captures certain information entered or saved by the user, with the corresponding threat to privacy. It causes the loss of information stored on the computer, either specific files or data in general. It affects the productivity of the computer, the network to which it's connected or other remote sites. It carries out actions that decrease the security level of the computer. It spreads, across the Internet. It redirects attempts to access web pages of certain banks to spoofed pages, with the aim of logging information entered by the user in these pages. In addition Delf.AAM, redirects attempts to access several web pages to a specific IP address.

2) Tiny.A

This is a Trojan, which can actually carry out attacks and intrusions: screen logging, stealing personal data, etc. It does not spread automatically using its own means. It needs an attacking user's intervention in order to reach the affected computer. The means of transmission used include, among others, floppy disks, CD-ROMs, email messages with attached files, Internet downloads, FTP, IRC channels, peer-to-peer (P2P) file sharing networks, etc. Tiny.E creates a winlogon launch point and runkey. It also modifies a number of registry keys.

3) W32.Sality.AE

It is a virus that spreads by infecting executable files and attempts to download potentially malicious files from the Internet. It uses the following infection strategies:

Resident: once it has been run, the virus goes memory resident and stops functions belonging to the operating system. Therefore, every time the operating system or an application tries to access any of these functions, the virus will activate and infect new files.

Stealth: when the virus is memory resident, it hides the modifications made to the boot sectors or files it has infected. These modifications are: changes in size, date, attributes, etc.

Encrypted: it encrypts its code in order to make it difficult to detect.

4) W32.Jeefo

W32.jeefo is a parasitic and memory resident virus that infects 32-bit Windows portable executable files. When an infected executable is run on the system, the virus will create the following file with the System attribute set: %Windir%\svchost.exe.

Its infection strategy consists in adding its code to the original file, increasing its size by 36,352 bytes. In addition, Jeefo encrypts the original file. Once an infected file is run, it is decrypted and disinfected.

5) Autorun

These worms spread by copying themselves into the root directories of hard drives and other writable media such as USB memory sticks. These worms create an autorun.inf file in the root directories of drives they want to infect. The autorun.inf includes the name and path of the actual worm executable. In addition to drives on the local computer, an Autorun worm can also spread to remote computers by infecting shared network drives.

6) Autorun.MG

It is a worm that spreads by copying itself, without infecting other files. It spreads and affects other computers. It uses stealth techniques to avoid being detected by the user. It spreads, via mapped drives, by infecting files that are then distributed. Autorun.MG uses the following propagation or distribution methods: Exploiting vulnerabilities with the intervention of the user, Computer networks (mapped drives) and File infection.

V. Evaluation measures

In order to check the ability of ANN to generalize and detect new malcode by employing fisher score's technique, we measured the *True Positive Rate (TPR)* measure, which is the number of *positive* instances classified correctly, as shown in (2), *False Positive Rate (FPR)*, which is the number of *negative* instances misclassified, as shown in (3), and the *Total Accuracy*, which measures the number of absolutely correctly classified instances, either positive or negative, divided by the entire number of instances shown in (4).

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

$$\text{Total Accuracy} = \frac{TP + TN}{TP + TN + FP + TN} \tag{4}$$

VI. Feature selection experiment and results

The aim of this experiment was to select best features [3], improve the performance of ANN as a classification method and minimize data capture's overload. We chose five different subsets of features ranked as cat5, cat10, cat20, cat50 and full. Each of the sets contained respectively 5, 10, 20, 50, 160 features. For each subset we performed six different experiments. In each experiment one of the six different malcode was removed

...

from the training set. During each experiment we trained the ANN using a dataset constructed from the examples of five malcode and 80% of the clean examples. The test set contained only samples of the sixth malcode, which did not appear in the training set, and an additional 20% of clean samples. Thus, the total number of experiments in this section was 5*6=30. The five features selected by fisher's score are presented in Table II. They are related to memory, system, IP.

TABLE II
Five features selected by fisher's score

1	*Memory-Free system Page table Entries*
2	*IP-Datagram Received Address Errors*
3	*System-System Up Time*
4	*Memory-System Code Total Bytes*
5	*System -% Registry Quota in use*

The malcode classification results using the fisher's score technique are summarized in Table III.

TABLE III
MALCODE CLASSIFICATION WITH FIHSER'S SCORE RESULTS

CAT5	*CAT10*	*CAT20*	*CAT50*	*FULL*	**Avg**
91	**85**	**81**	**80**	**59**	80

Each cell in table represents six different experiments, one for each missing malcode. The values in the cells are the detection accuracy averages of these six experiments. As shown above, the best accuracy was achieved by using only five features. "Fig. 1," represents trade off between the false positive and true positive rates.

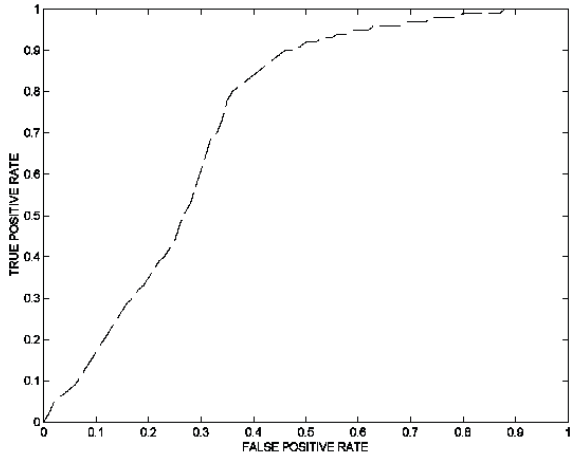

Fig.1. Trade off between the false positive and true positive rates

VII. CONCLUSION

In this paper, we increased honeypot security through data analysis by deploying ANN. We suggested employing the Fisher's score technique to capture efficiently the computer behavior in content of malcode activity.

We create and prepared a dataset for our ANN-model and our idea could decrease the risk of Honeypot. We evaluated accuracy of intrusion detection by ANN-model and our idea may increase the accuracy.

REFERENCES

[1] M. H López and C. F.Reséndez. "Honeypots: Basic Concepts, Classification and Educational Use as Resources in Information Security Education and Courses". Proceedings of the Informing Science & IT Education Conference (InSITE), 2008.

[2] D. Stopel, Z. Boger, R. Moskovitch, Y. Shahar and Y. Elovici." Application of Artificial Neural networks Techniques to Computer Worm Detection**".** 2006 International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006.

[3] R. Moskovitch, I. Gus, S. Pluderman, D. Stopel, C. Glezer, Y. Shahar and Y. Elovici. "Detection of Unknown Computer Worms Activity Based on Computer Behavior using Machine Learning Techniques," Department of Information System Engineering, Ben-Gurion University of the Negev 2006.

[4] A. Patcha * and Jung-Min Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends".accepted 9 February 2007 Available online , February 2007.

[5] D. Stopel, Z. Boger, R. Moskovitch, Y.Shahar, and Y. Elovici." Improving Worm Detection with Artificial Neural Networks through Feature Selection and Temporal Analysis Techniques".TRANSACTIONS ON ENGINEERING, COMPUTING AND TECHNOLOGY VOLUME 15 OCTOBER 2006.

[6] G. M Bednarski and J. Branson. "Information Warfare: Understanding Network Threats through Honeypot Deployment". Carnegie Mellon University. March, 2004.

[7] P. Picton, *Neural Networks*, 2nd ed., PALGRAVE, ISBN: 0-333-80287-X. New York, 2000.

[8] M. Hagan and M. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm, " *IEEE Transactions on Neural Networks*, Vol. 5, No. 6,, pp. 989-993, November 1994

[9] *S. Mardovich*. "Network packet payload analysis for intrusion detection". ".accepted 9 February 2007 Available online February 2007.

[10] Honeynet Project (2003) Know Your Enemy: Honeynet. Retrieved March , http://www.honeynet.org/papers/honeynet/index.html., 2006.

• • •