Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 4 Issue 10, October - 2017

Remote Operating System Identification Using Artificial Neural Networks

O. O. Obe, B. K. Alese
Department of Computer Science, Federal University of Technology
P. M. B. 704, Akure, Nigeria.

M. A. Oduwale
Department of Computer Science, Ondo State University of Science and Technology
P. M. B. 353, Okitipupa, Nigeria.

***Abstract*—The need to determine the identity of an Operating System is valuable in areas such as network security, internet modeling, and end to end application design. Various techniques have been used to solve the problem of remote operating system identification such as rule-based tools (e.g. network mapper). However, these techniques fail to detect the operating system with high accuracy. In this study, multilayer perceptron model that utilizes back propagation neural network based classifier was developed for accurately fingerprinting operating system of remote host. This network was simulated using Neural Designer simulator. The result of this study was compared with Network mapper and machine learning technique using accuracy. The result comparison proved that Back Propagation neural network based classifier is far more accurate than rule-based tools on packet traces for fingerprinting.**

Keywords—Feed Forward, Back Propagation, Feed Forward Neural Network, Operating System, Neural Designer

I. Introduction

Remote Operating System identification systems popularly known as operating system (OS) Fingerprinting, is an essential part of the assessment process of a network security. Operating system (OS) fingerprinting is the process of determining the identity of the Operating System of a remote host on the internet. This may be accomplished passively by sniffing network packets travelling between hosts, or actively by sending carefully crafted packets to the target machine and analyzing the response, it leverages the fact that different operating systems implement differing TCP/IP stacks, each of which has a unique signature. Even between versions or patches of an operating system there exist subtle differences as developers include new features and optimize performance [11].

The operating system fingerprinting is a process of remotely detecting and determining the identity of a target system by observing the TCP/IP packets that are generated by that system [1].

Operating system fingerprinting, also known as TCP/IP stack fingerprinting, is the process of determining the operating system of a target system based on inferring properties of its TCP/IP protocol stack from observed packets [5].

Robust and practicable operating system (OS) Fingerprinting must meet some requirements. At first, it must be accurate, i.e. it does not fingerprint the operating system (OS) falsely; secondly, it must be quick for allowing large network scans; furthermore, it also need that the signature database can be extended easily. To meet these requirements, the design of the classifier in the OS tools plays an important role.

Operating system fingerprinting is the process of learning what operating system is running on a particular device. By analyzing certain protocol flags, options, and data in the packets a device sends onto the network, we can make relatively accurate guesses about the operating system (OS) that sent those packets. By pinpointing the exact operating system (OS) of a host, an attacker can launch a precise attack against a target machine.

Ultimately, most of the researchers have derived various tools for fingerprinting operating system. In this research, the artificial neural networks concept was used using neural designer simulator to fingerprint correctly.

1.1. Related Studies

[6] used Passive Operating System Identification from TCP/IP Packet Headers to fingerprint.

New classifiers were developed using machine-learning approaches including cross-validation

testing, grouping OS names into fewer classes, and evaluating alternate classifier types.

[1] used Machine Learning Techniques to improve Operating System Fingerprinting. TCP/IP communication is setup between machines to capture and analyze TCP/IP packets for more accurate and fine grained OS detection using packet correlation approach was built.

[4] utilized support vector machine to classify OS fingerprint. An evaluation of using a support vector machine (SVM) to classify operating system fingerprints in the Nmap security scanner. In solving a simplified version of operating system classification, the SVM got marginally more accurate results than Nmap's built-in classifier.

[9] used Machine Learning Techniques for Advanced Passive Operating System Fingerprinting. This paper focused on automating the generation and updating of the signatures for passive fingerprinting. It deals with fingerprints which do not have an exact match with an already known signature using classification algorithms.

[10] developed a new tool and technique for remote operating system fingerprinting. This work present an original Operating System detection method, based on temporal response analysis.

2. Methodology

We studied the technical variables from the packet header file of the operating system. We selected twelve input variables from the signature datasets.

We trained a three-layered feed-forward neural network model with backpropagation algorithm. Output values were recorded and compared with the target.

The model was simulated using Neural Designer simulator.

3. Design and Implementation

We applied Neural Designer Simulator to enhance modular techniques in the design of basic functional components. There are six main functional modules of the fingerprinting system (Fig. 1). These include the data gathering module, input phase, neural network module, performance functional, training strategy module and the output module.

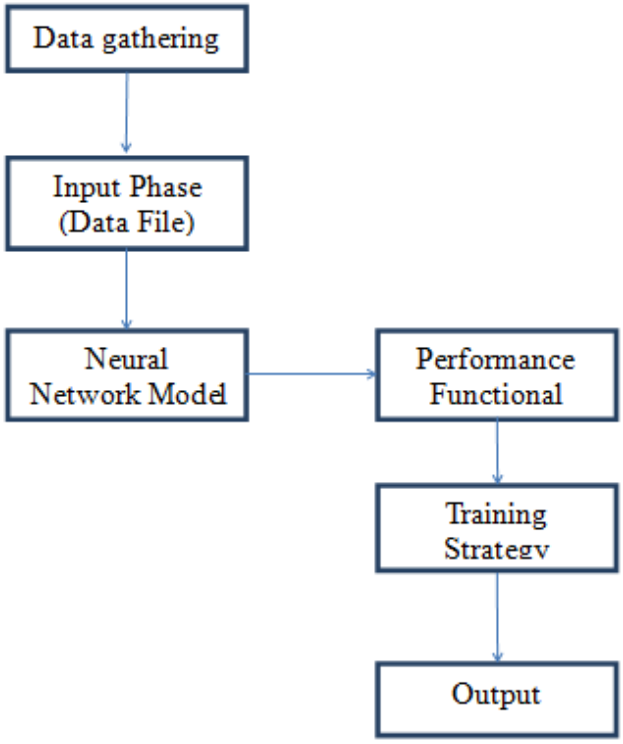

Fig 1. *Conceptual diagram of the neural-network-based fingerprinting software*

At one end, datasets signatures were gathered. Numeric formats were submitted to the input phase. The digital signature datasets were thereafter submitted to the neural network module as input signals for training the network or testing as applicable. The neural network module had the features to save the network configuration for trained networks for subsequent testing and to submit the result of the tested input to the output module wherein it could be displayed or printed. The conceptual diagram of the system is shown in fig. 1.

3.1. Setting up the Network

3.1.1 Task Description

The neural network defines a function which represents the model. A neural network is defined within Neural Designer as a multilayer perceptron. It is a class of universal approximator. This neural network is used to span a function space for the variational problem at hand.

3.1.2 Data Set

The first step is to prepare the data set, which is the source of information for the remote operating system identification problem.

The pOf.dat file contains the data for this example. In this example we have a data set with 13 variables (columns) and 100 instances (rows).

3.2. Training Strategy

The next step in solving this problem is to assign the training strategy, which is composed of two different algorithms:

- Initialization algorithm
- Main algorithm

3.3. Running the Network

The training data were used to run the network.

4. Results

A Neural Network produces a set of outputs for each set of inputs applied.
The outputs depend, in turn, on the values of the parameters.

Table 1 shows the input values and their corresponding output values. The input variables are X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11 and X12; and the output variable is X13

Table 1: Input values and their corresponding output values

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13
1	128	8.19e+003	0	6	1	1	0	3	1	1	1	0	0.168
2	255	0	7	0	1	0	0	1	1	1	1	0	0.994
3	128	8.19e+003	0	2	1	1	0	3	1	1	1	0	10.9
4	128	8.19e+003	0	8	1	1	0	3	1	1	1	0	0.168
5	64	6.55e+004	0	4	1	1	1	3	1	1	1	1	10
6	64	0	10	6	1	1	1	1	1	1	1	0	0.0311
7	64	0	20	10	1	1	1	1	1	1	1	0	0.105
8	128	6.55e+004	0	1	1	1	0	3	1	1	1	0	0.005
9	128	8.19e+003	0	6	1	1	0	3	1	1	1	0	0.168
10	64	1.64e+004	0	0	1	0	0	0	0	0	0	0	5.99

The following deductions can be derived from the inputs-outputs table

1. X13 is the output data
2. 0.005 – 0.168 indicate a varying range of Windows operating system versions
3. 5.99 - 10.9 indicate a varying range of Unix operating system versions

5. Conclusions

This research demonstrated the applicability of training data to ANN for solving selection problem. The artificial neural network has been able to establish a useful method of accurately fingerprinting operating system.

This research utilized a three-layer feed forward neural network model trained with back propagation learning algorithm.

Though training may be tedious and expensive, it is opined that if a network is well trained with a proper set of input data, artificial neural networks will generate better results than other traditional techniques.

REFERENCES

[1] Al-Shehari, T. and Shahzad, F. (2014). Improving Operating System Fingerprinting using Machine Learning Tecniques. International Journal of Computer Theory and Engineering, Vol. 6, No. 1. pp. 1 – 6.

[2] Burroni, J. and Sarraute, C. (2005). Using Neural Networks for remote OS Identification; In Proceedings of the Pacific Security Conference (PacSec '05), Tokyo, Japan, November 15-16.

[3] Chris, T. (2011). An Overview of Remote Operating System Fingerprinting". URL:http://www.sans.org/reading_room/whitepapers/testing/overview-remote-operating-system-fingerprinting_1231.pdf (12 March 2015).

[4] Fifield, D. (2010). OS Fingerprint classification using a support vector machine

[5] Greenwald, L. G and Thomas T. J (2007). Toward undetected Operating system fingerprinting. In Proceedings of the First USENIX Workshop on Offensive Technologies (WOOT'07), Boston, MA.

[6] Lippmann, R., Fried, D., Piwowarski, K., and Streilein, W. (2003). Passive Operating System Identification From TCP/IP Packet Headers**.** In Proceedings of the ICDM Workshop

[7] Negnevitsky, M. (2005), Artificial Intelligence - A Guide to Intelligent Systems, 2nd Edition, Pearson Education Inc., Essex, England.

[8] Sarraute, C., and Burroni, J. (2008).Using Neural Networks to improve classical Operating System Fingerprinting techniques; Electronic Journal of SADIO, Vol. 8, No. 1, pp. 35–47. Retrieved from: http://www.coresecurity.com/files/attachments/SarrauteEJS.pdf

[9] Schwartzenberg, J. (2010). Using Machine Learning Techniques for Advanced Passive Operating System Fingerprinting URL: eprints.eemcs.utwente.nl/18789/01/Julius_-_Final_version.pdf. (15 January, 2015).

[10] Veysset, F., Courtay, O., Heen, O. (2002). New Tool and Technique for Remote Operating System Fingerprinting. Intranode Research Team. v1.1

[11] Wenwei, L., Dafang, Z., and Jinmin, Y. (2005). Remote Operating System Fingerprinting Using BP Neural Network.