Parameter Significance for Email Attack Detection

Khaled Alduhaiman
School of Information Tech.
George Mason University
Fairfax, VA 22030
kalduhai@gmu.edu

Michael Francis
George Mason University
4789 Walbern Ct
Chantilly, VA 2003
rfrancis2@Cox.net

Sultan Aljahdali
School of Information Tech.
George Mason University
Fairfax, VA 22030
saljahda@gmu.edu

ABSTRACT
High speed network traffic monitoring generates large volumes of data. This data is composed of parameter sets that are extracted from packets (and packet switching processes) as they transit points of inspection. Effective countermeasures and event detection capabilities for early attack sensing and warning (ASW) are predicated on methods that can be deployed at an autonomous system boundary in the carrier backbone network. Thus the optimal point of inspection (for ASW) is in a very noisy, high-speed / large-data volume environment. Inspection and detection schemes are severely constrained by the speed-volume factors and the low information content (quality) that is available at this point of inspection.

1. INTRODUCTION

A hidden Markov modeling (HMM) technique to detect denial of service/ distributed denial of service attacks required parameters for the basic model. The initial effort has focused on the question: " what observable parameters are important?" To answer this question, researchers developed some techniques to evaluate the significance of parameters. The general objective is to find the minimum set of parameters needed to detect specific events. The ultimate intent is to produce efficient algorithms for the detection of events in high- speed gateways where minimal knowledge is available concerning the source and/or destination computing environments.

2. METHODOLOGY: GROUNDED THEORY

Grounded theory method (GTM) is a process for the construction of theory from observable phenomena. With GTM, the theory is inductively derived from the study of the behavior under consideration. There are three basic elements of grounded theory: concepts, categories and propositions. Grounded theory method is used in this research to develop theory concerning the observable features of browser requests directed to a Web Server. The intent is to inductively derive a generalized relationship between the observable features that identifies the browser request as either an attack (or part of an attack) or a valid browser request. There are five analytic phases (and not strictly sequential) in this process [Pandit, 1996]:

i. Research design- defines the data capture method, conceptualization of data and the categorization technique.
ii. Data collection- at the Web Server, observation is constrained to the features that are observable in the browser request.
iii. Data ordering- by browser request type (i.e. know attack, know valid, etc.)
iv. Data analysis - apply mathematical algorithm, perform statistical analysis and data mining
v. Literature comparison- review of related research

The grounded theory method is well-suited to this problem. The researchers could not develop a theory *a priori* and then test it. In fact, it was not known that any theory actually existed. The grounded truth method provides an iterative process where data collection, analysis and theory stand in a reciprocal relationship [Strauss, Corbin 1990] Specifically for this research, it was assumed that if any theory existed, a systematic collection, management and analysis of data pertaining to the observable phenomena would yield an inductively derived construction of the theory. This method assumes that the variables have acted and the research is limited to measuring the effects. [In counterpoint a true experiment manipulates variables and measures the causal effects]. A weakness of this method is the requirement to:

- Infer a hypothesis (or set of hypotheses) from the statistical analysis.
- Eliminate competing hypotheses, until a single hypothesis is validated [Hicks, 1993].

This paper presents some initial findings and does not attempt to identify and eliminate competing hypotheses in a formal proof. The limits of this work in its current state are readily acknowledged. However, we have noted that a serious effort is underway by many organizations to create systems that can provide ASW functionality. We believe a structured approach to build a framework for rigorous analysis and proof of hypothesis is needed for these efforts. This paper provides an introduction to several concepts that apply.

The *CaptureNet* tool extracted packet feature information from the point of inspection depicted below to build the

required data sets. The data was then stored in comma separated value (CSV) format for replay and statistical analysis. The victim machine is an email server, the attack machine is a *Microsoft Windows 2000* client, running email attack programs. The "normal" mail traffic features are extracted from a machine that is assume to have normal hygiene (i.e. the traffic is free of any type of email attack).

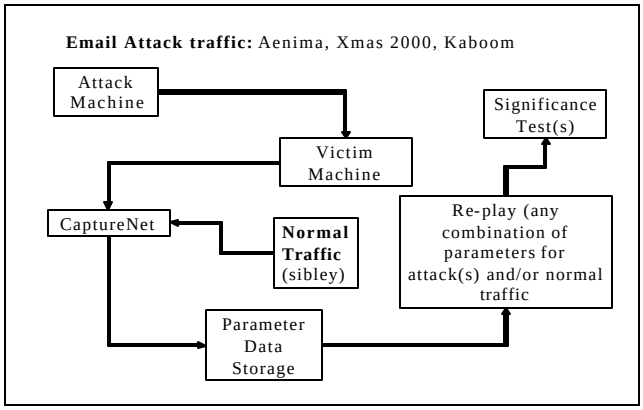


Figure 1. Lab configuration

3. ANALYSIS

A neural network was used to evaluate parameter significance. *PathFinder™* [Z Solutions, Inc.] is an adaptive neural network that learns using an iterative process called back propagation. A discussion of neural networks is beyond the scope of this paper, but the interested reader is referred to a tutorial available from Z Solutions for additional information. The following five parameters were selected for significance testing: (X1) source port, (X2) destination port, (X3) sequence number, (X4) acknowledgment number and (X5) packet size. The sigmoid transformation was selected for the output, where A (output = 0) is a determination of an attack and N (output =1) is a determination that the traffic is normal. Figure (2) depicts the configuration of the *PathFinder™* Three sets of data were generated:

1. Training set- uses parameters derived from normal (sibley) traffic and two attack programs (Aenima, Xmas 2000).
2. Test set- uses non-overlapping parameters from the same traffic types (sibley, Aenima and Xmas 2000) as the training set.
3. Validation set- uses non-overlapping data from the previous traffic types (sibley, Aenima, Xmas 2000) plus the addition of another attack traffic parameter set (Kaboom).

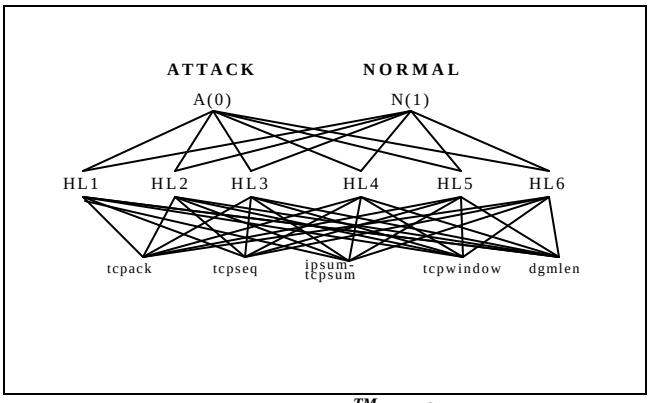


Figure 2. *PathFinder™* configuration

Pathfinder™ is an application that was designed to help users utilize neural networks. It has an interface with *Microsoft Excel* for data management and analysis. The learning algorithm employs back propagation. *Pathfinder™* uses three data sets to perform a neural network analysis. The three data sets are the (1) training set, (2) the test set, and (3) the validation set.

- The **training set** is the set of data used by the neural network to learn the problem.
- The **test set** is used during training to keep an eye on learning performance.
- The **validation set** is used after training as a final check to find out how well the model performs.

The learning parameters in a neural network control the rate of learning. *Pathfinder™* uses an advanced algorithm to set the learning parameters for our problem. The network architecture (shown in figure 2) is controlled by these parameters:

- **Output nodes**- the number of output nodes, the number of output nodes is one and the result either 0 or 1.

- **Output transform function**- the output transformation function *Pathfinder™* allows two possible output transformations: the sigmoid transformation or the linear transformation. Z Solutions recommends using sigmoid transformation.
- **Hidden Nodes**- The number of hidden nodes controls the number of weights in the model. Usually, the greater the complexity of the problem the more nodes are needed. On the other hand, too many nodes may lead to over-fitting. *Pathfinder™* defaults to 6 hidden nodes and rarely requires significantly more.
- **Epoch size**- The epoch size is the number of observations seen by the learning algorithm before weight adjustments. The idea is to look at a large enough epochs that noisy extraneous data points will not excessively influence the results and small enough that specific details can be

determined. *Pathfinder™* defaults to an epoch size of 12. This is a good place to start and usually experiment in the range of 6 to 64.

The data considered in this study was collected in three scenarios. Each email attack tool was instructed to bomb one hundred emails from its machine to the target. After recording all the emails traffic at the target computer, packets were averaged based on their email size. Number of packets' email varies depending on their size. If one email has 10 packets, they were averaged to one packet. Moreover, one hundred normal emails were sent from their machines to the victim's machine. Seven different email-bombing tools were used and four different normal emails were sent. Four of the email bombing were mixed with two of the normal emails and used during the training of the neural networks. Also, two of the email bombing were mixed with two of the normal emails and used during the validation process.

Significance testing was performed on several parameters:

- Datagram length (dgmlen),
- Acknowledgment number (tcpack)
- Sequence number (tcpseq)
- Header checksum for both TCP and IP (tcpsum , ipsum)
- Window size (tcpwindow).
- A class variable - was added to show whether the email was normal (1) or attack (0).

The inputs are dgmlen, tcpack, tcpseq, tcpsum, ipsum and the output is class. The data is divided into three data sets: training data, test data, and validation data. Training and test data were collected first and validation data were collected second. The validate data is used after training is completed to determine how accurate a neural network model is on data not seen during training. Samples of training, test and validation data are shown in the following tables.

Table 1. Sample training data

dgmlen	ipsum	tcpseq	tcpack	tcpwindow	tcpsum	class
659.8	26844.7	2508747052	3067442870	11042.4	43778.3	**1**
577.5	26699.6	2508999080	3067682224	14101	26189.3	**1**
59.3	30288.9	2732277838	2318855137	17135.7	28384.9	**1**
75.1	33540.9	1385924998	1385925020	12596.7	33503.9	**0**
110	31719.8	2648319293	2648319336	16723	32863.5	**1**
308.9	29614.2	2457418608	3016359125	16564.7	32257.8	**1**
756.2	25863.8	2229478565	3346824468	12798.1	23301.9	**1**
902.2	25437	2229698073	3346918666	12809.1	21216.9	**1**
106.5	20534.5	4104389228	4104389272	16891.8	32608.4	**0**
223.2	28027.8	677676449	677676478.2	17215.9	40316.8	**0**
756.2	26002.6	2229373722	3346779126	13514.3	28012	**1**
916	26488.3	2508810165	3067484471	14067	28188.2	**1**
236.5	33311.8	2965863799	2847922313	17357.8	39297.6	**1**

Table 2. Sample of test data

dgmlen	ipsum	tcpseq	tcpack	tcpwindow	tcpsum	class
78.2	28127.8	677683544	677683570.3	17081.7	33021.5	**0**
222.1	28138.9	677658924	677658957.3	17192.8	36687.3	**0**
127.5	20388.5	4104398779	4104398848	16745.7	35787.2	**0**
720.4	29169.4	2153211985	3271149311	14657.5	33739.3	**1**
223.2	28252.8	677640504	677640532.9	17257.1	31643.9	**0**
94.7	20763.3	4104372507	4104372546	16666.1	30355	**0**
377.1	30626.4	2991676034	2432713351	13896.5	18562.3	**1**
75.1	21975.9	1385919206	1385919228	12448.2	25574.9	**0**
126.5	20871.5	4104361689	4104361757	17008.2	40494.3	**0**
223.7	28476.3	677605269	677605298.8	17249.5	33677.2	**0**
728	28724.9	2182284131	3300163111	16516.9	25981.1	**1**
169.6	31521.1	2656060032	2656060059	17185.1	37282.7	**1**
86.3	20801.7	4104370149	4104370180	17438.5	40441.5	**0**

Table 3. results (highlighted were missed)

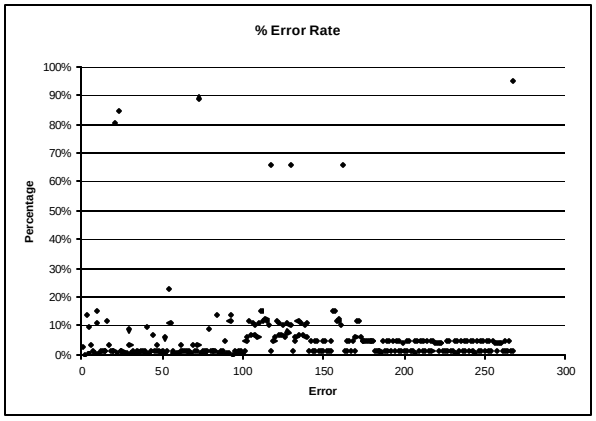
dgmlen	ipsum	tcpseq	tcpack	tcpwindow	tcpsum	class	Predicted
116.6	28815	3067703904	2508976443	16926.4	24227.6	1	0
75.1	19405.9	1385928859	1385928881	12452.7	32479.4	0	0
122.5	20358.5	4104401495	4104401559	16884.4	40971.5	0	1
85.3	28260.7	677661404	677661438.8	16896.7	27233	0	0
987.6	25516.2	2229557428	3346858449	13525.3	28613.6	1	0
916	26131.3	2509059837	3067650919	12963.4	37750.6	1	1
105.9	20747.1	4104372927	4104372970	16390.5	38828.5	0	0
75.1	25830.9	1385918153	1385918175	12017.7	28111.4	0	0
95.7	18942.9	3808333236	3808333278	16577.2	40127.1	0	0
75.1	10410.9	1385922365	1385922387	12265.2	32546.9	0	0
745.2	26307.3	2509055093	3067647818	11106.8	19659.4	1	0
126.5	20434.5	4104395301	4104395367	16949.2	41833.9	0	0
805.8	28693.6	2177965057	3295836986	13299.1	29807.3	1	1

Table 4. Sample of validation data

dgmlen	ipsum	tcpseq	tcpsequp	tcpack	tcpackup	tcpw indo w	tcpsum	class
720.4	33693.1	1864463456	28448.8	2193325658	33467.2	14538.7	23395.3	?
226.5	36018.5	2296906199	35047.3	2296906241	35047.3	16927.9	37351	?
175.5	36101.5	2296653235	35043.7	2296653251	35043.7	17420.8	40386.6	?
209.4	41104.6	2022477647	30860	2022477940	30860	17076.1	41528.6	?
304.6	37518.3	1946667757	29703.4	2111102747	32212.6	16592	38595.7	?
79.1	36231	2296486259	35041	2240366093	34184.8	17074.2	38180.1	?
350.7	40935.8	2024539357	30891.5	2024539773	30891.5	16665.2	38980.1	?
805.8	33618.9	1864454013	28448.8	2193321939	33467.2	16516.9	21188.9	?

The data considered in this study were obtained by using the three scenarios discussed above. The first test results suggest that it is possible to distinguish between normal and attack email streams with more than 85 % of accuracy, as shown in Figure 3. The table below shows a sample of these results.

Figure 3. Error Rates, showing an 88 % Accuracy



The highlighted rows are false alarms (i.e. mismatches between the class and the predicted columns). Four different error calculations were performed on the test results. The error calculations are defined as follows:

MAE - Mean absolute error

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|Forcast_i - Target_i\right|$$

Where N = Number of observations (i. e. the number of rows in the data matrix).

MSE - Mean square error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Forcast_i - Target_i)^2$$

Where N = Number of observations.

The mean square error calculates the square of the errors;

RMSE - Root mean square error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{Forcast}_i - \boldsymbol{Target}_i)^2}$$

Where N = Number of observations.

Where N = Number of observations.

MAPE-Mean absolute percentage error

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{Forcast_i - Target_i}{Target_i}\right|$$

Table 5. Errors values

Training Data	400 rows
Test Observations	233 rows
Validation Observations	193 rows
RMSE	0.3274
MAPE	3.27 %
MAE	0.5

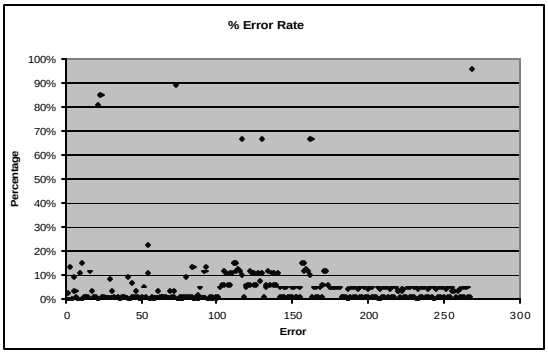


Figure 4 Error vs. Predicted

4. RESULTS WITH NEURAL NETWORK

RMSE - Root mean square error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{Forcast}_i - \boldsymbol{Target}_i)^2}$$

Where N = Number of observations.

The root mean square error is simply the square root of the mean square error.

MAPE-Mean absolute percentage error

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{Forcast_i - Target_i}{Target_i}\right|$$

Where N = Number of observations.

During validation testing, *PathFinder™* correctly associated (100%) of the parameters used in the training (and test) data sets with either a normal or attack traffic event as appropriate. Parameters associated with a previously unseen attack ("Kaboom") were correctly associated with an attack 85% of the time.

Data

	Correct	Incorrect
Attack	138 (83.1%)	28 (16.9%)
Normal	27 (100%)	0 (12.9%)
	165 (85.5%)	28 (14.5%)

5. REFERENCES

[1] Young, Steve, (et al.) *The Hidden Markov Modeling Tool Kit (HTK) Book*, version 3.0, Microsoft Corporation, July 2000.

[2] Zalewski, Michal, *Strange Attractors and TCP/IP Sequence Number Analysis*, white paper, BindView Corporation, Houston Tx., www.bindview.com, 2001.

[3] Z Solutions, *Pathfinder Neural Network System: a tutorial*, Z Solutions LLC, Atlanta, Ga. www.zsolutions.com, 1998.

[4] Elliott, John., Distributed Denial of Service Attacks and the Zombie Ant Effect, IT Pro, March –April 2000, pg 55-57

[5] Comerford, Richard., *No Longer in Denial*, IEEE Spectrum, January 2001, pg 59-61.

[6] Geng, Xianjun., Whinston, Andrew.B., *Defeating Distributed Denial of Service Attacks*, IT Pro, July-August 2000, pg 36-41.

[7] Schuba, Christoph.L., Krsul, Ivan.V, Kuhn,Markus.G., Spafford,Eugene.H., Sundaram,Auobindo., Zamboni,Diego., *Analysis of a Denial of Service Attack on TCP*, COAST Laboratory, Department of Computer Sciences, Purdue University, IEEE 1997, pg 208-223.

[8] Berry, Michael J.A., and Linoff, Gordon. *"Data Mining Techniques"*, Wiley 1997 P 287.

[9] Aleksander,I.;Evans,R.G.;Sales,N. *Towards intentional neural systems*: experiments with MAGNUS 1995., Artificial Neural Networks, Fourth International Conference on , 1995 Page(s): 122 –126

[10] Amoroso Edward, Intrusion Detection, an introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response. Intrusion.Net Books, June 1999.

[11] Northcutt, Stephen, *Network Intrusion Detection Ana Analysis Handbook*, New Riders Publishing, Indiana, June 1999.

[12] Ptacek, Thomas and Newsham, Timoth. *Insertion, Evasion, and Denial of Service*: Eluding network Thomas intrusion detection. Technical Report, Secure Network, Inc., January 1996.

[13] Rabiner L., "A tutorial on Hidden Markov Models and selected applications in speech recognition", 1989, Proc. IEEE 77(2):257--286.

[14] Simonds, F. Network Security: Data and Voice Communications. McGraw-Hill, 1996.

[15] Stallings, W. *Network Security Essentials*; Prentice Hall: 1996-366