Generating Representative Traffic for Intrusion Detection System Benchmarking

H. Güneş Kayacık, Nur Zincir-Heywood

Dalhousie University, Faculty of Computer Science,
6050 University Avenue, Halifax, Nova Scotia. B3H 1W5
{kayacik,zincir}@cs.dal.ca

Abstract

In this paper, a modeling and simulation framework is proposed for generating data for training and testing intrusion detection systems. The framework can develop models of web usage from web server logs in a data driven fashion and the actual traffic is generated by employing the web browser installed on the host. Additionally, we employed an intrusion detection system as a traffic analyzer to validate the synthetic data that the framework generated and compared it against the standard intrusion detection system benchmark data, namely KDD 99 datasets.

Keywords

Network Security, Intrusion Detection, Traffic Modeling, Markov Models, Self-Organizing Maps

1 Introduction

Being a crucial part of security management operations, intrusion detection systems should be tested thoroughly to determine and eliminate the weaknesses. Intrusion detection system benchmarking involves tests on a mixture of attacks and normal behavior to evaluate detection capabilities as well as false alarms on normal behavior. Generating attacks and testing detection rate is fairly easy as compared to the modeling of normal behavior. Attack data is sufficient to test signature based intrusion detection systems because they simply aim to match the attack signatures with the input data. Therefore normal usage can be simplified as the background noise from which intrusion detection system distinguishes the signals (i.e. attacks). On the other hand, in case of anomaly detection systems, normal behavior is learnt from the training data therefore the definition of normal behavior plays an important role in the quality of the intrusion detection system. The objective of our recent research is to develop an intrusion detection system benchmarking framework, which can develop models of normal activity from observations, and generate synthetic data based on normal activity models. Given the objectives, we believe that the developed framework will be useful to researchers who work on network simulations or intrusion detection system design to conduct repeatable experiments.

The remainder of the paper is organized as follows. Section 2 describes the framework for synthesizing data. Section 3 provides details of a Self-Organizing Map based intrusion detection system, which we will employ as a traffic analyzer. Results are presented in Section 4. Conclusions are drawn in Section 5.

2 Data Synthesis

Although a comprehensive benchmark should cover wide range of protocols, we focus on modeling one protocol, namely hypertext transfer protocol (HTTP). HTTP is selected because considerable amount of the Internet traffic is made up of HTTP traffic [1]. Although our framework is implemented for HTTP, it could easily be tailored for many other protocols, which involve file transfers with variable inter-arrival times.

2.1 Data Source

Two basic resources exist for modeling web usage: server side data and client side data. Server side logs are automatically generated by web servers and contain the URL requests for web pages, images and objects located at a web site. Each request contains information about the timestamp, client address and resource being requested. Additionally it can contain optional information such as client browser type and referrer page.

On the other hand, client data is obtained by collecting the browsing habits of users who participate in browsing tests. Client side data is harder to obtain because of the privacy issues and the people participating should use a special purpose browser to record the sessions. To avoid such obstacles, server side data, which is collected from Ege University Vocational School web server, is employed in this work. Characteristics of the web server log are detailed in Table 1. In this work, definition of normal behavior is considered as the nominal use of a web server, which is summarized in terms of URL requests in web server logs. Web server logs can contain anomalous requests, which are crafted by hackers to exploit the potential web server vulnerabilities. Before modeling, we ensured that the web server log is free from such anomalies.

Table 1. Characteristics of the Web Server Log of Ege University Vocational School

Year	Duration	# of Requests	# of Distinct html pages
2004	1 month	43994	132

Since HTTP is not a session-based protocol, web server logs only contain the records of requested documents without any information about where a session ends. Session concept is imperative because it acts as a placeholder to group the activities of a user. Session boundaries can be determined based on IP addresses and time stamps. Empirical tests proposed by He et al. [2] determines session boundaries by examining the inter-arrival times of URL requests. The assumption is that URL requests that belong to the same session will have short inter-arrival times, whereas long inter-arrival times will mark the end of a session. According to the empirical test results in [2], if the inter-arrival between two subsequent requests from the same IP address is longer than 15 minutes, it marks the end of a session. We applied the same empirical tests to Ege Vocational School web server log and determined that 5 minutes mark the end of a session.

2.2 Developing Models

In this work, we want to model the web page transitions as well as the inter-arrival times. In order to model the transition probabilities between requests, first order discrete Markov Models [3] are employed. Models of inter-arrival times are developed by a modified first order discrete Markov Model.

2.2.1 Markov model for web page transition

In a first order Markov model, next state depends on only the current state. A Markov model is defined by the state space, I; transition probability matrix, P; and the probability distribution of the states, λ.

Let I be a countable set of states, where $i, j \in I$ represent states. If there are N states in I, a first order Markov model can be represented as a two-dimensional $N \times N$ matrix $P = (p_{ij} \mid i, j \in I)$ where p_{ij} is probability of moving from state i to j. Let X_t define the state at step t. From the training data, probability of transition from state i to j can be calculated as follows;

$$p_{ij} = \frac{C(X_t = j, X_{t-1} = i)}{\sum_{i} C(X_{t-1} = i)}$$

$C(X_t = j, X_{t-1} = i)$ is the number of times state j follows state i in the training data. The i^{th} row of the P is the probability distribution of moving to all states from state i. This probability distribution is also called λ_i.

2.2.2 Markov model for transition delay

In continuous-time Markov models [3], a rate matrix Q is introduced to control the rate of transition from i to j. The transition delay between states i and j are exponentially distributed with the rate parameter q_{ij} In this paper we adopt a different approach by employing discrete-time model over continuous-time. As opposed to exponential distribution assumption in continuous-time Markov models, our use of discrete time Markov models does not make any assumptions about the underlying probability distribution. That is to say, probability distributions are formulated based on the training data. In our approach, similar to continuous-time Markov models, transition delay counts are stored in a rate matrix Q.

Let $D \subseteq [0, \infty)$ be a set of transition delays observed in training data. Extending the discrete Markov model, we define a three-dimensional matrix $Q = (q_{ijd} \mid i, j \in I, d \in D)$ where q_{ijd} is the probability of moving from state i to j with d time step delay. Similar to the continuous-time Markov models, Q matrix controls the transition time between states, except rather than fitting the observed data to exponential distribution, it allows formulation of arbitrary distributions. Hence, q_{ijd} can be calculated as follows;

$$q_{ijd} = \frac{C\left(X_t = j, X_{t-1} = i, \Delta(t, t-1) = d\right)}{\sum_{i,j} C(X_t = j, X_{t-1} = i)}$$

$\Delta(t, t-1)$ is the transition delay between state t and $t-1$. $C\left(X_t = i, X_{t-1} = j, \Delta(t, t-1) = d\right)$ is the number of times state j follows state i with a delay of d. When i and j are fixed ($q_{ijd} \mid i \in I, j \in I$) is a probability distribution of transition delay for the transitions from i to j.

2.2.3 Modeling methodology

In this research, each web page is considered as a state. Images are discarded because they are requested when the enclosing web page is requested. Similarly, dynamic pages such as *cgi* are omitted because in addition to the document name, parameters passed to a dynamic web page determine content. Moreover, since dynamic web pages might update databases, the traffic that we generate might lead to unwanted changes in databases during our tests. Additionally, two special purpose states are introduced, namely start-state and end-state. Each session starts from the start-state and is terminated with the end- state. This way, the model can simulate infinite number of session lengths. Simply, new requests will be generated until the end-state is reached. Moreover, rather than starting from a random state, probability distribution of the start-state indicates which pages are more likely to be a starting point. Transition delays in Q matrix are expressed in terms of seconds.

2.3 Generating Synthetic Data

Transition probabilities in matrices *P* and *Q* from Markov Models are utilized to generate sessions. Each new session starts from the start-state. Next page is selected stochastically utilizing the probability matrix *P*. In the stochastic selection process, the greater the probability in web page transition matrix *P*, the greater the chances that web page will be visited. After the next page is selected, a suitable inter-arrival time is selected stochastically utilizing the probability matrix *Q*. Stochastic selection for the session continues until the end state is reached. To employ in our experiments, we generated 10000 synthetic sessions. Since it is desired to have an experimental framework, where replication of results is possible, synthetic sessions are stored in a file hence separating session generation from traffic generation.

Traffic is generated between our test machine and the web server from which the web server log was obtained. Generating traffic involves processing synthetic sessions to send the URL requests with corresponding inter-arrival times but more importantly processing the answer returned by the web server. That is to say, if the requested page contains images or frames, they should also be requested. Therefore, as opposed to developing a small agent to send URL requests, the web browser that is installed on the machine is employed to generate the traffic. This way, behavior of web browsers can also be incorporated to the traffic generation process. This method also reduces the chance of creating simulation artifacts. In our experiments, we employed Mozilla version 1.4 installed on a Linux machine. A script reads in the synthetic sessions and interacts with the browser to generate traffic.

3 Data Analysis

In order to determine the robustness of the generated synthetic dataset, we employed a Self-Organizing Map based intrusion detection system [4] as a network traffic analyzer to compare the characteristics of synthetic data and real-world data. The objective of the analysis is to determine whether our synthetic dataset shows improvements over the standard dataset –namely KDD 99 dataset –, which is based on MIT Lincoln Lab's DARPA intrusion detection evaluation datasets.

3.1 KDD 99 Dataset

Given the significance of the intrusion detection problem, there have been various initiatives [5, 6, 7] that attempt to quantify the current state of the art. MIT Lincoln Lab's DARPA intrusion detection evaluation datasets stand out from other initiatives and have been employed to design and test intrusion detection systems. These datasets, to the best of authors' knowledge, are the most comprehensive undertaking in intrusion detection system benchmarking. In the DARPA intrusion detection evaluation initiative, a simulation is made of a factitious military network consisting of three 'target' machines running various operating systems and services. Additional three machines are then used to spoof different IP addresses, thus generating traffic between different IP addresses. System calls were recorded from the target machines and the traffic was recorded by a sniffer which can see all network traffic. The total simulated period is seven weeks. Normal connections are created to profile that expected in a military network and various attacks were initiated to exploit different services.

In 1999, recorded network traffic from the DARPA 98 Lincoln Lab dataset [7] was summarized into network connections. This formed the KDD 99 intrusion detection dataset in the International Knowledge Discovery and Data Mining Tools Competition [8]. Each network connection is characterized with 41 features where first 6 features are basic features, which can be derived from packet headers without inspecting the content. These basic features are as follows:

- Duration of the connection;
- Protocol type, such as TCP, UDP or ICMP;
- Service type, such as FTP, HTTP, Telnet;
- Status flag, derived by *Bro* to describe a connection;
- Total bytes sent to destination host;
- Total bytes sent to source host;

To derive 41 features per connection, a customized *Bro* intrusion detection system was employed [9]. This customized version is not publicly available furthermore the publicly available version [10] can only derive 6 features per connection. Therefore, in our analysis, we employed 6 basic features from KDD datasets and employed publicly available Bro to summarize *tcpdump* data into connection records with these 6 features.

3.2 Data Source

For data analysis, we employed two synthetic datasets: (1) 10% KDD, the original training set in KDD 99 competition (2) Synthetic data that we generated from Ege University Vocational School Web Server Logs. Additionally we employed a real-world dataset, which is the captured traffic from Dalhousie University Faculty of Computer Science web server Locutus. Since our synthetic dataset only contains HTTP connections, other datasets are filtered to contain only HTTP connections. Three SOM hierarchies are trained by using one dataset for training and the remaining two for testing.

3.3 Two-Level SOM Hierarchy

In our previous work [4] a two level SOM hierarchy, which is shown in Figure 1, was developed for intrusion detection. Self-Organizing Map [11] is a neural network algorithm, which employs unsupervised learning for training. Compared with the related research [9, 12, 13, 14], which employ 41 features of a connection, the main difference of this work lies in utilizing only the 6 basic features of a connection, which can be derived from packet headers without inspecting the packet payload. At the first level, six SOMs are trained, one for each feature where the general objective is to encode temporal relationships. The second level combines the information from the first level SOMs. In order to facilitate detection, second level neurons are labeled by using the training set labels after the training is completed. Thus, the training process itself is unsupervised. Other than the representation of temporal features and labeling of second level neurons, no further *a priori* information is employed.

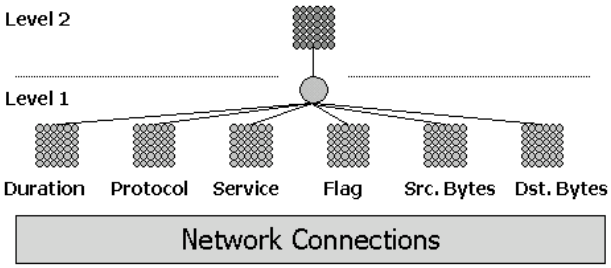

Figure 1. Two-Level SOM hierarchy

3.4 Data Analysis with SOM Hierarchy

Various papers have been published which establish the shortcomings in the benchmarking procedures [15] and in the generated data [16] in DARPA benchmark initiatives. More specifically, it is found that the attributes of the synthetic data varies over a narrow range, whereas the real-world data has a wide range of variation and contains faulty packets containing bad checksums, malformed arguments or IP fragmentation [16]. Although these differences may have fewer implications for signature based intrusion detection, it is imperative to include accurate models of normal behavior to be able to distinguish abnormal behavior. Otherwise it would be impossible to tell whether a connection is an attack or merely an unfamiliar behavior that was not encountered in the training data. Therefore, in order to develop a machine learning based intrusion detection system that can withstand in real-world environments; it is crucial to synthesize training data that has the similar characteristics to real-world data. To this end, we employed two level hierarchical SOM for the analysis of synthetic and real-world datasets in this work.

Since real-world datasets are unlabeled, detection and false alarm rates cannot be calculated. Thus, hit histograms are employed to analyze datasets. Hit histograms summarize which SOM neurons are excited (i.e. selected as the best matching neuron) for a given dataset. As a neuron is excited for more patterns in a dataset, the area of the hexagon being colored becomes larger. In these experiments, SOM hierarchy acts as a clustering method to summarize the characteristics of the training dataset. In essence, the SOM explores by stretching in the search space. If the training and the test data have similar statistical properties such as similar range, probability distribution and dispersion, then the test data would populate a considerable portion of the SOM.

4 Results

Figures 2, 3 and 4 contain the hit histograms of three SOM hierarchies, which are trained with different datasets. In each of these figures, one hit histogram belongs to the training set and the other two belong to the test sets.

The hierarchy trained on 10% KDD (Figure 2) demonstrates that mainly one neuron is excited for the given data. This suggests that the organization of the SOM trained on 10% KDD is unable to distinguish the characteristics of any dataset other than its training set (to a minimal degree).

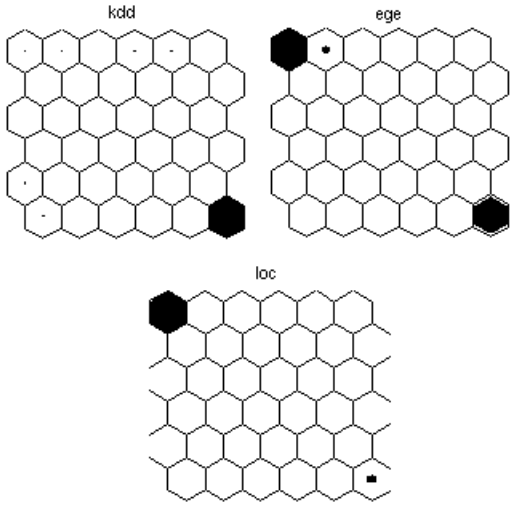

Figure 2. Data hit histograms on SOM Hierarchy trained on 10% KDD HTTP data.

The system trained on our synthetic data (Figure 3), which is based on Ege University Vocational School web server log, showed improvements compared to the system trained on 10% KDD. In Ege SOM hierarchy, patterns in real-world dataset locutus start to excite multiple neurons. This indicates that the organization of the SOM hierarchy

can comparatively distinguish different patterns in real-world dataset.

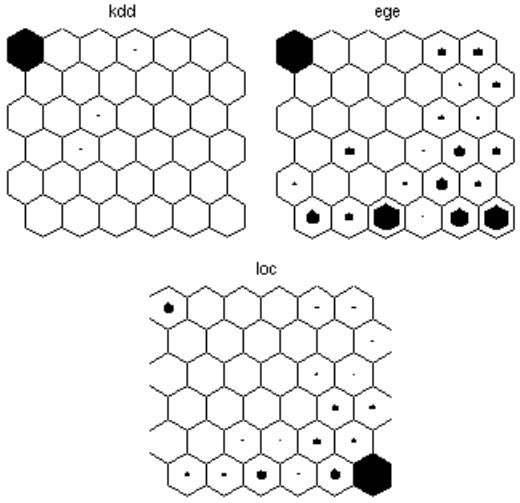

Figure 3. Data hit histograms on SOM Hierarchy trained on our synthetic HTTP data, which is named as Ege.

In order to determine how the SOM hierarchy would be organized when trained on a real-world data set, Locutus data is employed for training as well. In case of Locutus SOM hierarchy, Locutus and Ege datasets excited multiple neurons whereas 10% KDD mainly excited only one neuron, Figure 4.

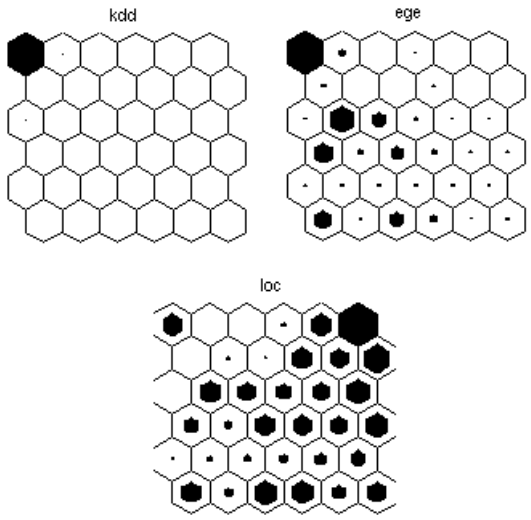

Figure 4. Data hit histograms on SOM Hierarchy trained on Locutus HTTP data.

5 Conclusions and Future Work

In this paper, we presented a modeling and simulation framework, which can develop models of web usage from web server logs. Models are developed from the web server logs without assuming any underlying probability distributions hence modeling framework is data driven. The set of components that we developed within the proposed framework comprise a comprehensive toolkit, which can develop usage models and generate traffic. By providing analytic modeling and facilitating repeatable simulations, this framework fills a gap in intrusion detection benchmarking.

In order to validate the synthetic data that the framework generates, we compared the synthetic data with the standard dataset 10% KDD by using a SOM based intrusion detection system as a network analyzer. Given that the SOM aims to achieve topological arrangement with respect to the training set, if the training and the test data have similar characteristics, the test data would excite a substantial portion of the SOM. Experimental results showed that the synthetic data generated by the framework shows improvements over 10% KDD dataset in terms of being more similar to the real-world data.

The future work will investigate the use of other datasets from commercial and governmental organizations to develop models of usage and the extension of the framework to other protocols such as FTP. Possible protocols would involve file transfers with variable request times.

References

[1] Odlyzko A., “Internet traffic growth: Sources and implications”, 2003, http://www.dtc.umn.edu/~odlyzko/doc/itcom.internet.growth.pdf, Last accessed Mar. 2005.

[2] He, D., & Göker, A. (2000), “Detecting session boundaries from Web user logs”, In Proceedings of the BCS-IRSG 22nd annual colloquium on information retrieval research, Cambridge, UK (pp. 57-66).

[3] Norris J.R., Markov Chains, Cambridge University Press, 1997, ISBN: 0-521-48181-3

[4] Kayacik, G. H., Zincir-Heywood, A. N., Heywood, M. I., “On Dataset Biases in a Learning System with Minimum a Priori Information Intrusion Detection”, Proceedings of the IEEE CNSR, Fredericton, Canada, May 2004.

[5] Debar, H., Dacier, M., Wespi, A. and Lampart, S. 1998. “An experimentation workbench for intrusion detection systems”, Res. Rep. RZ 2998 (#93044) (Sept.). Research Division, IBM, New York, NY

[6] Puketza N.J., Zhang K., Chung M., Mukerjee B., "A Methodology for Testing Intrusion Detection Systems," In IEEE Transaction on Software Engineering v 22 no 10 (Oct 1996) pp 719- 729 http://citeseer.ist.psu.edu/puketza96methodology.html.

[7] The 1998 intrusion detection off-line evaluation plan. MIT Lincoln Lab., Information Systems Technology Group.

http://www.11.mit.edu/IST/ideval/docs/1998/id98-eval-11.txt, 25 March 1998.

[8] Knowledge discovery in databases DARPA archive. Task Description. http://kdd.ics.uci.edu/databases/kddcup99/task.html

[9] Lee W. and Stolfo S. "A Framework for Constructing Features and Models for Intrusion Detection Systems", Information and System Security, 3(4): 227–261, 2000.

[10] Paxson V., "Bro: A System for Detecting Network Intruders in Real-Time", Computer Networks, 31(23-24), pp. 2435-2463, 14 Dec. 1999.

[11] Kohonen T., Self-Organizing Maps. Springer, Third edition, 2001. ISBN: 3-540-67921-9

[12] Pfahringer B. "Winning the KDD99 Classification Cup: Bagged Boosting" SIGKDD Explorations, 1(2):65–66, 2000.

[13] Eskin E., Arnold A., Prerau M., Portnoy L., and Stolfo S. "A Geometric Framework for Unsupervised Anomaly Detection: Detecting intrusions in unlabeled data", In Applications of Data Mining in Computer Security. Kluwer, 2002. ISBN 1-4020-7054-3, 2002.

[14] Joshi M.V., Agarwal R.C., and Kumar V. "Mining needle in a haystack: classifying rare classes via two-phase rule induction", SIGMOD Record (ACM Special Interest Group on Management of Data), 30(2):91–102, 2001.

[15] McHugh J., "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory", ACM Transactions on Information and System Security, Vol. 3 No.4 November 2000.

[16] Mahoney M.V., Chan P.K. "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection", In Recent Advances in Intrusion Detection, 6th International Symposium, RAID 2003, Pittsburgh, PA, USA, September 8-10, 2003, Proceedings. Lecture Notes in Computer Science 2820 Springer 2003, ISBN 3-540-40878-9.