Höglund A. J., Hätönen K. and Sorvari A. S., 2000, A Computer Host-Based User Anomaly Detection System Using the Self-Organizing Map, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), vol. 5, pp. 411-416.

A COMPUTER HOST-BASED USER ANOMALY DETCTION SYSTEM USING THE SELF-ORGANIZING MAP

Albert J. Höglund, Kimmo Hätönen, Antti S. Sorvari
Nokia Research Center,
P.O. Box 407, FIN-00045 NOKIA GROUP, Finland
albert.hoglund@nokia.com, kimmo.hatonen@nokia.com, antti.sorvari@nokia.com

Abstract
Computer systems are vulnerable to abuse by insiders and to penetration by outsiders. The amount of monitoring data generated in computer networks is enormous. Tools are needed to ease the work of system operators. Anomaly detection attempts to recognize abnormal behavior to detect intrusions. A prototype UNIX Anomaly Detection System has been constructed. The system is host-based and monitors computer network host users. The system contains an automatic anomaly detection component. This component uses a test based on the Self-Organizing Map to test if user behavior is anomalous. Both the test and the application are presented in this paper.

Keywords: Self-Organizing Map (SOM), anomaly detection, novelty detection, host-based intrusion detection

1. Introduction

Computers and computer networks are becoming more and more important. The computer networks are normally protected from unauthorized usage by security mechanisms, such as passwords and access controls. However, if an abuser or intruder manages to bypass these security mechanisms and gains access to vital information, the potential loss is enormous.

Detecting intruders or abusers at an early stage can decrease the loss. The two main intrusion detection techniques are rule- or signature-based misuse detection and anomaly detection. Rule-based misuse detection attempts to recognize specific behaviors that are known to be improper. That is, if a user follows certain intrusive patterns, he is classified as an intruder. Anomaly detection, on the other hand, attempts to recognize anomalous or abnormal user behavior to detect intrusions. Anomalous or abnormal behavior is suspected if the current behavior deviates sufficiently from the previous behavior, which is assumed normal.

Lunt gives a good overview of intrusion detection techniques in [10]. Statistical anomaly detection has been used for intrusion detection e.g. in NIDES [12]. The references [1, 2, 6, 9] also provides interesting information on anomaly detection and signature-based intrusion detection. Anomaly detection has the same objectives as novelty detection, i.e. detecting novel or abnormal behavior. Tarassenko uses a Gaussian Mixture Model (GMM) in [13] for detection of masses in mammograms. A problem with the GMM is the large number free parameters to be estimated when the amount of training data is limited. In [11] Nairac et al. successfully uses a novelty detection algorithm based on K-means clustering to detect anomalous jet engine vibrations. This algorithm is quite close to the algorithm presented in this paper. The main difference is that the SOM test for Anomaly is based on the Self-Organizing Map (SOM), that it calculates anomaly P-values and reports the most abnormal features.

The amount of monitoring data generated in computer networks is enormous. The objective was to construct an Anomaly Detection System to ease the work of system operators. The system should automatically detect abnormal computer network user behavior and provide means for behavior analysis.

A prototype Anomaly Detection System [3, 4, 5] was constructed for the UNIX environment. The system consists of a data gathering component, a user behavior visualization component, an automatic anomaly detection component and a user interface. The user behavior visualization component uses large SOMs to visualize the user behavior. It was presented in detail in [4]. This paper is focused on the automatic anomaly detection component and the SOM test for anomaly. Also the UNIX Anomaly Detection System in general will be briefly presented in this paper.

0-7695-0619-4/00 $10.00 © 2000 IEEE

2. The SOM Test for Anomaly

2.1 The Concept

The objective with the SOM Test for Anomaly is to test if the current behavior of an object is anomalous or not. The hypothesis to be tested is:

H_0: The most recent observation is not anomalous.
H_1: The most recent observation is anomalous.

The behavior of an object can be very consistent, which means that it is concentrated to one or a couple of regions in the feature space. It can on the other hand also be more scattered in the feature space, which would signify a more irregular behavior. The idea of the SOM Test for Anomaly is to approximate the normal behavior of an object with a small object specific SOM [7]. The previous behavior is assumed to represent the normal behavior of the object. Anomalous observations can be omitted from the previous behavior when training the SOMs.

The behavior of a one-dimensional SOM is illustrated in Figure 1. 200 points of artificial data for two features have been plotted in the plane together with the neurons of a map of size 8*1 trained with the data. The one-dimensional SOM approximates the two clusters of data quite well. Note that the map in Figure 1 have been made using two-dimensional data in order to explain the method. In a real situation the number of features is not limited.

The Best Matching Unit (*BMU*) for a data point $\mathbf{f}_k$ in a SOM is the neuron $\mathbf{w}_i$ that lies closest to the data point. This is expressed in (1), where d stands for the distance measure. In this application the Euclidean distance to the *BMU* is used to measure how much an observation deviates from the normal object specific behavior. The Anomaly P-Value is a measure of the degree of anomaly for an observation. On the basis of this value the hypothesis H_0 is rejected or accepted. The Anomaly P-value is calculated according to the algorithm below.

2.2 The Algorithm

1. A set of features describing the object is selected. The feature vector describing the object is denoted by $\mathbf{f}$.
2. The normal behavior of the object is observed. This means that n measurements ($\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n$) of the feature vector is collected.
3. A SOM with m neurons is trained, using the measurements ($\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n$) as training data. The number of neurons in the map, m, is selected to be much smaller than n, for example $n/10$.
4. Neurons in the SOM that are not Best Mapping Units for any of the data points ($\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n$) are omitted.
5. The Best Mapping Unit -distances for ($\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n$) are calculated from the SOM trained in step three. These distances are denoted by ($D_1, D_2, \ldots, D_n$).
6. The objective is to test the most recent observation $\mathbf{f}_{n+1}$ for anomaly. The hypothesis to be tested is H_0: The most recent observation $\mathbf{f}_{n+1}$ is not anomalous. The alternative hypothesis is H_1: The most recent observation $\mathbf{f}_{n+1}$ is anomalous.
7. The *BMU* distance for the observation $\mathbf{f}_{n+1}$ is then calculated. This distance is denoted by D_{n+1}.
8. Let B be the number of the Best Mapping Unit distances ($D_1, D_2, \ldots, D_n$) that are bigger than D_{n+1}. The Anomaly P-value for a certain object is then calculated as in (2).

$$BMU = \underset{i}{\operatorname{argmin}}\{d(\mathbf{f}_k, \mathbf{w}_i)\}, \qquad P_{n+1} = \frac{B}{n} \tag{1), (2}$$

9. If the Anomaly P-value is bigger than the Anomaly P-value threshold, the null hypothesis H_0 is accepted. If, on the other hand, the Anomaly P-value is smaller than the Anomaly P-value threshold, H_0 is rejected, that is the most recent data point is assumed anomalous.
10. If the test indicates that the object behavior is anomalous (H_0 is rejected), the k most significantly deviating features can be determined. The k features (components of the feature vector) with the biggest absolute contribution to the *BMU* distance are the k most significantly deviating features. Equation (3) shows how the most deviating feature is calculated. This component of the feature vector is given the sub-index *md* in equation

(3). In equation (3) *BMU* stands for the Best Mapping Unit of the feature vector $\mathbf{f}_{n+1}$ and j takes values from zero to the number of features. The other k-1 most deviating features are calculated in a corresponding manner.

$$f_{n+1,md} = \underset{j}{\text{argmax}}\{abs(f_{n+1,j} - BMU_j)\} \quad (3)$$

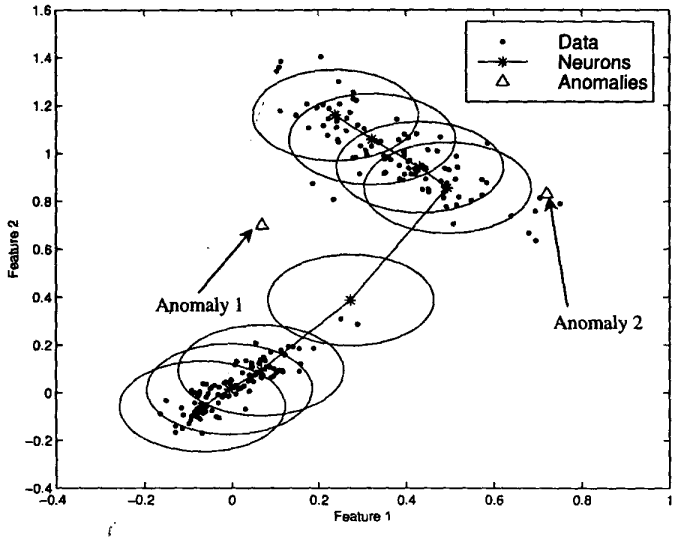

*Figure 1 A one-dimensional (8*1) SOM trained with 200 points artificial data. On the left circles have been plotted using the neurons of the two-dimensional SOM as centers.*

The situation shown in Figure 1 can be used as an example. The Anomaly P-value for Anomaly 1 is 0.00 (0/200). Since zero of the BMU-distances for the data points have a BMU-distance greater than that of Anomaly 1, the value of the numerator is zero. Correspondingly the Anomaly P-value for Anomaly 2 is 0.035 (7/200).

2.3 Alarm Levels

If the Anomaly P-value is smaller than the Anomaly P-value threshold, H_0 is rejected and an alarm is triggered. The Anomaly P-value threshold can be interpreted as the fraction of observations that will be rejected if the behavior of the monitored object does not deviate from the earlier behavior of the same object. That is, if the null hypothesis is true the number of alarms is *P-value threshold * observations*. On the other hand, if the null hypothesis is not true (i.e. the new data is anomalous), the number of rejections (alarms) increases.

A selected P-value threshold can be illustrated for object i using d-dimensional spheres centered at the neurons of the object specific map. Here d stands for the number of dimensions for the training data ($\mathbf{f}_1$, $\mathbf{f}_2$, . . . , $\mathbf{f}_n$). The number of observations for object i that fall outside the spheres correspond to the numerator B in (2). The two-dimensional example in Figure 1 illustrates the situation. Here B is 13, which correspond to a quite high P-value threshold of about 6.50.

3. The UNIX Anomaly Detection System

3.1 System Overview

The UNIX Anomaly Detection System monitors the daily behavior of computer network host users. Figure 4 illustrates the four components of the system. The routines that build the UNIX Anomaly Detection System have been coded using C and Perl and the SOMs are trained using the procedures of SOM_PAK [8]. The user interface html-based. For details on the user behavior visualization component see [3, 4].

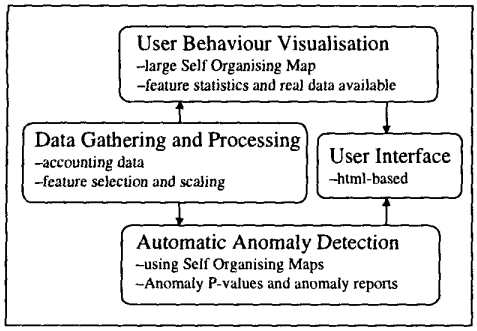

Figure 2 Overview of the Anomaly Detection System for UNIX.

3.2 Data Gathering, Feature Selection and Scaling

The system operates in a UNIX environment. The data gathering and the data processing are performed on separate servers. This enhances security and makes it more difficult to disturb the operation of the system. For a period of 1100 days the user account logs of more than 900 users have been stored.

Feature selection provides a means of reducing the enormous amount of data generated by computer network users. Features describing the object must be selected in such a way that it is possible to detect the desired anomalies or deviations in the object profile. Features characterizing the user behavior during a period of 24 hours were used in the UNIX Anomaly Detection System. Firstly an initial feature set with 34 features was derived. Features describing CPU-time and different services were included, but also session, process and login information. The initial feature set was reduced to a set of 16 features. This was achieved by omitting features with strong linear dependency to other features and by omitting very noisy features. Careful consideration and expert knowledge was used in the feature selection process, though.

Since the magnitude of the features varies greatly, logarithmic or linear normalization was used. Histograms and variances were studied to find an appropriate scaling factor and normalization method for each feature.

3.3 Automatic Anomaly Detection

The automatic anomaly detection component relies on the SOM Test for Anomaly. Here each user corresponds to an object to be analyzed. The period of reference or the period on which the user profile is based is normally the previous 90 days. The number of measurements or feature vectors, n_i, for each user i can be different. In this case n_i is the number of active days during the period of reference for the user i. Only active behavior is tested for anomaly. The number of neurons in the user specific SOM is selected to be $m_i=n_i/10$.

Anomaly reports are produced daily. In these reports anomalous user behavior for the previous day is reported. In these reports the Anomaly P-value, the level of activity and the three most abnormal features for the reported user are listed. The system provides links to graphs, statistics and raw data for analysis of the user behavior reported anomalous.

Figure 3 provides an anomaly detection example based on real data. In this figure the behavior of a single user is analyzed. A short training period of 30 days is used in this example. The first training period is located on the left-hand side of the vertical straight line located at *time* 30 days. The system is retrained using the previous thirty days before analyzing a new day. The scaled feature values are plotted as a function of time (a constant was added to each feature for better visualization). A P-value threshold of 0.05 is used. The dashed vertical lines in Figure 3 are reported anomalies. The three most abnormal features for the anomalies are presented in Table 1.

The reason for the anomaly on day 31 seems to be that feature values of ROOT, PROCESSCOUNT and CPU11-17 are clearly higher than normal. The same applies for the anomaly on day 33 but now also the feature SERVICE4 has been used (not used previously by the user). These features are also reported by the system as the most abnormal features (see Table 1). The reason for the anomaly on day 52 is that the usage of CPU has been higher than normal. On day 57 and 69 the length of the processes has been longer than normal. The reason for the anomaly on day 69 seems to be a combination of a high value in CPU17-23 and a bit lower value in PROCESSCOUNT than normal for this amount of CPU.

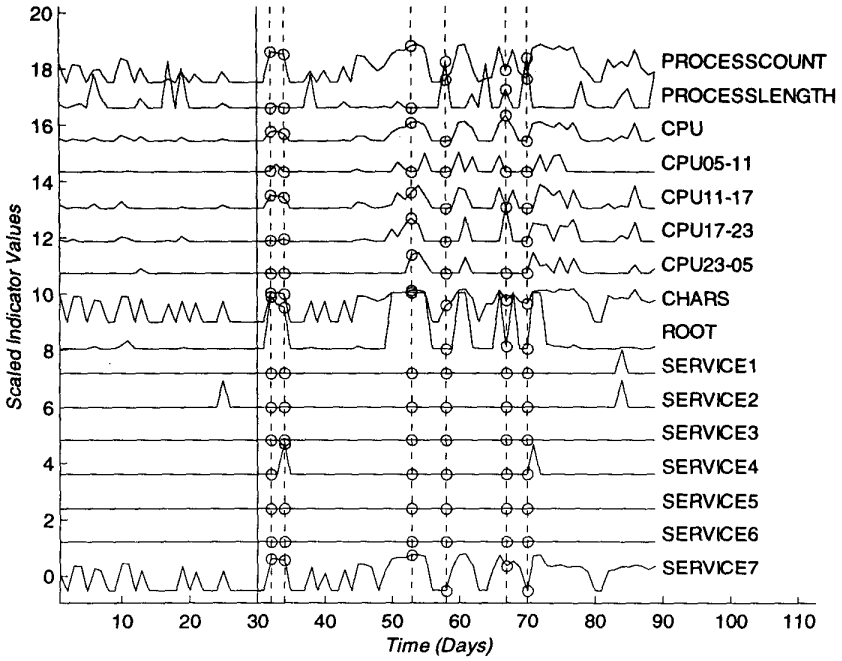

Figure 3 Anomaly detection feature graph for the examined user.

These kinds of anomalies are interesting for the system administrator (e.g. higher usage than normal of ROOT, anomalous usage of services etc.). A longer training period gives more reliable Anomaly P-values. This means that a lower P-value threshold can be used, and that the report number of anomalies will be lower.

Table 1 Most abnormal features for the anomalies of the user in Figure 3.

	Day 31	Day 33	Day 52	Day 57	Day 66	Day 69
1. Most Abnormal Feature	ROOT	SERVICE4	CPU23-05	PROCESS-LENGTH	CPU17-23	PROCESS-LENGTH
2. Most Abnormal Feature	CPU11-17	PROCESS-COUNT	CPU	SERVICE5	PROCESS-COUNT	SERVICE6
3. Most Abnormal Feature	PROCESS-COUNT	ROOT	CPU17-23	SERVICE6	CPU	SERVICE5

4. Conclusions and Comments

The prototype Anomaly Detection System for UNIX is in test usage. The feedback from the test usage has in general been quite positive. Comments like "This system reduces the amount of data to be analyzed" and "The system gets

you when you are doing something suspicious" have been encouraging. The link to the real data has also been found excellent. And the system has also found several interesting anomalies. The system does of course also produce false positives or false alarms, which means anomalies that are not intrusions. Since the system is an anomaly detection system, normal changes in the user profiles may lead to false positives. For example when a user suddenly changes to a different type of project.

The system gradually adapts to changes in the user profiles. This means that it also adapts to intrusive behavior. Therefore the response to intrusive actions should be immediate. The longer the period of reference used when training the user specific SOMs of the System, the slower the system adapts to changes in the user profiles. Also the length of the time interval between training of the user specific SOMs affects the system's speed of adoption. In general the prototype UNIX Anomaly Detection System with the SOM test for Anomaly provides an excellent way to reduce the amount of data to be analyzed.

As a final conclusion some interesting properties of the SOM test for anomaly:

- Adopts to changing data
- Detects a feature or variable that is out of its' normal range
- Detects abnormal feature or variable combinations (also those when individual features not out of range)
- Can handle different kinds of distributions and combinations of them, multi-cluster, etc.
- The method is multivariate
- The method also reports the most abnormal features or variables

5. References

[1] EMERALD homepage: http://www2.csl.sri.com/emerald/index.html, 1999.

[2] Frank J., Machine Learning and Intrusion Detection: Current and Future Directions. Proceedings of the National 17th Computer Security Conference, 1994.

[3] Höglund, A. J. An Anomaly Detection System for Computer Networks. Master's Thesis. Helsinki University of Technology, 1997.

[4] Höglund, A. J. and Hätönen, K., Computer Network User Behavior Visualization using Self-Organizing Maps, in: Proceedings of ICANN 98: 899-904, 1998.

[5] Höglund, A. J., Hätönen, K., Tuononen T. O., A UNIX Anomaly Detection System using Self-Organizing Maps, Workshop on the Recent Advances in Intrusion Detection (RAID'98), 1998, http://www.zurich.ibm.com/~dac/RAID98.html.

[6] ISS homepage: http://www.iss.net/, 1999.

[7] Kohonen, T., Self-Organizing Maps. Second Edition. Springer-Verlag, Heidelberg 1997.

[8] Kohonen, T., J. Hynninen, J. Kangas, J. Laaksonen. SOM_PAK, The Self-Organizing Map Program Package, Version 3.1, April 1995.

[9] Kumar S, Spafford EH. A pattern matching model for misuse intrusion detection. In Proceedings of the 17th National Computer Security Conference, October 1994, pp. 11-21.

[10] Lunt T. F. A survey of intrusion detection techniques, Computers and Security 1993; 12(4): 405-418

[11] Nairac, Townsend, Carr, King, Cowley and Tarassenko. A system for the analysis of jet engine vibration data. Integrated Computer aided engineering 6: 53-65, 1999.

[12] NIDES homepage: http://www2.csl.sri.com/nides/index.html.

[13] Tarassenko, Hayton, Cerneaz, and Brady. Novelty Detection for identification of masses in mammograms. In the proceedings of 4th IEE Conference on ANN, 117-122, 1995.