**Automated Anomaly Detection Within The Toa Network Flow Data Monitoring System**

José Alfredo Valles, Dr. José Ortiz-Ubarri

Department of Computer Science, University of Puerto Rico-Río Piedras Campus, San Juan, Puerto Rico, 00931

**Abstract**

Detecting network anomalies can help protect sensitive data, prevent attacks, monitor network performance, and strengthen network security. Toa is an open source, pseudo-real time network monitoring system (NMS) that provides an easy to deploy web interface for system and network administrators to monitor high volumes of network traffic. Anomaly detection through Toa is done with visualization, and the current work aims to develop an algorithm to optimize and automate the process of anomaly detection with adaptability to the context and situation, integrating this new feature into Toa. To this end, we implemented statistical algorithms that use distinct time ranges, and an exponential smoothing algorithm to facilitate anomaly detection.

**Introduction**

The key objective of this project is to develop an automated network anomaly detection algorithm capable of adapting to the network behavior within context and situation. For the purpose of our project, an anomaly is defined as anything that deviates from the normal behavior of the network, such as:

- network malfunctions
- network attacks.

This will be integrated into the Toa NMS, allowing it to monitor anomalies automatically. The Toa system stores the network flow information within a database, and each data entry is an aggregate of the:

- input and output octets
- input and output packets
- number of flows

of the networks that are being monitored. This allows for easy access, making it simple and time-efficient to examine large portions of the data to continuously define network behavior models. Since detecting anomalies, as well as establishing network behavior models is an important and difficult research topic, we believe the contributions in this area will be important with this project.

**Methodology**

Using a script with queries written with Python and SQL, we have examined multiple time-ranges, including

- the past week,
- past month,
- past three months,
- every five minutes for five months,
- among others.

For now, we have tested a simple algorithm using the data’s standard deviation and a simple operation to calculate a threshold. If the data point exceeds that threshold, it is considered an anomaly.

1. for i in range ID:
2.   oneDay(ID)
3. for i in range oneDay:
4.   if oneDay[i] > thresh:
5.     print alert

Figure 1: Pseudocode for basic anomaly detection algorithm using every facility ID within the Toa database, calculating the network flow data for the past day.

We are currently concentrating on smoothing the data to test if it is a more efficient way to separate the prominently abnormal data points, as well as defining a sliding window model to allow the algorithm to adapt to changes within the network, and constantly redefine an anomaly.

1. Figure 2: Smoothed Data vs. Original Data

**Results**

This method has proved successful in detecting anomalies with the various time ranges and facility IDs monitored by Toa. We have tested the data with multiple experiments, and the time ranges used return the correct results, making the development of future algorithms and models easier. Data smoothing has proved successful for now, but further tests with the current defined anomaly detection algorithm are necessary.

1. Figure 3: Example of Anomaly Detection

**Future Work**

Further test the anomaly detection algorithm with real and experimental data; further refine the network behavior model by examining various machine learning approaches; examine the use of tree decision structures, combining specific and context-aware conditions to accurately detect anomalies; examine the use of clustering techniques to partition a set of data into groups with similar characteristics, labeling any sample that does not belong to a cluster as an outlier; examine the use of a majority vote system where alerts are triggered only if the majority of the accurately tested algorithms detect an anomaly.

**References**


**Acknowledgements**

The authors would like to thank the members of the CSLab and the Computer Science Department, as well as the CAHSI Summit for their support.

This work is partially supported by the National Science Foundation under Grant No. DUE-1438838. Presentation of this work has been supported in part by NSF Grant CNS-1042341.