Bayesian Detection of Router Configuration Anomalies

Khalid El-Arini
Computer Science Department
Carnegie Mellon University
kbe@cs.cmu.edu

Kevin Killourhy
Dependable Systems Laboratory
Computer Science Department
Carnegie Mellon University
ksk@cs.cmu.edu

ABSTRACT

Problems arising from router misconfigurations cost time and money. The first step in fixing such misconfigurations is finding them. In this paper, we propose a method for detecting misconfigurations that does not depend on an *a priori* model of what constitutes a correct configuration. Our hypothesis is that uncommon or unexpected misconfigurations in router data can be identified as statistical anomalies within a Bayesian framework. We present a detection algorithm based on this framework, and show that it is able to detect errors in the router configuration files of a university network.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network management

General Terms

Management, Reliability

Keywords

Statistical anomaly detection, Router configuration

1. INTRODUCTION

On January 23, 2001, Microsoft's websites went down for nearly 23 hours. The next day, Microsoft spokesman Adam Sohn attributed the failure to a "configuration change to the routers on the DNS network" [4]. This example highlights the critical problem of router misconfiguration. Since companies rely on the availability of their networks, such misconfigurations are costly. Each router is individually configured with its own *router configuration file*, which can contain several thousand lines of commands. While the syntactic correctness of each file can be verified, determining semantic correctness and consistency across all the router configuration files in a network is a much harder problem. Due to the magnitude of this problem, our goal is to develop a method that automatically identifies semantic mistakes among the set of router configuration files that define each network.

Previous approaches [2, 3] to this problem require an *a priori* expectation of what these configuration files should look like. Our work differs in that we make no such assumptions about the structure of router configurations. It is inspired by a suggestion made by Caldwell et al. [1]. They noted that network design choices might be inferred by commonalities across router configuration files, and that these might be learned automatically and codified as rules. Our work does not directly employ a rule learner, but our algorithm detects such misconfigurations as statistical anomalies just the same.

SIGCOMM'05 Workshops, August 22–26, 2005, Philadelphia, PA, USA.
ACM 1-59593-026-4/05/0008.

2. DETECTION ALGORITHM

A joint Bayes detection algorithm was designed and implemented to test our hypothesis.[1] The algorithm has a training phase and a detection phase. The training phase examines each line of every configuration file and computes a set of key frequencies describing the commands and their arguments. The detection phase makes a second pass through the files, using these frequencies to find anomalies.

Our algorithm makes the simplifying assumption that each line of a configuration file is independent of every other line. However, it does not assume that the attributes of a single command are independent of one another. Consider a line L in a configuration file that consists of a command c and attributes $(a_1, a_2, \ldots, a_n)$, where commands are defined as Cisco IOS keywords and attributes are the remaining whitespace-delimited words on the line. This algorithm estimates the probability of the line given the command $P(L \mid c)$ as the joint probability of all the attributes given the command: $P(L \mid c) = P(a_1, a_2, \ldots, a_n \mid c)$.

During the training phase, the algorithm estimates these probabilities as follows. For each line L with command c and attributes $a_1, a_2, \ldots, a_n$, the probability of the line given the command is estimated as the fraction of instances of the command c that contain the entire sequence of attributes a_1 through a_n. If we use $\#(c)$ to denote the number of times command c appears and we use $\#(a_1, a_2, \ldots, a_n \mid c)$ to denote the number of times the sequence of attributes appears for command c, then the probability $P(a_1, a_2, \ldots, a_n \mid c) = \frac{\#(a_1, a_2, \ldots, a_n \mid c)}{\#(c)}$.

In the detection phase, we calculate the probability of a line $P(L_i \mid c)$ using these estimates. We determine whether or not the line is an anomaly by comparing this probability to a threshold. A general threshold across all commands in the configuration file would not identify anomalies. Consider the example of two commands, c_1 and c_2. Each command appears 24 times, and each takes a single argument. The

[1]The algorithm was named "joint Bayes" to differentiate it from two other Bayesian detection algorithms that were also investigated, but omitted due to space constraints.

command c_1 appears once with argument x_1 and 23 times with argument x_2. The command c_2 appears once with each argument y_i, for $i \in \{1, \ldots, 24\}$. The lines "c_1 x_1" and "c_2 y_1" both have equal probability of occurring (one in 24). However, "c_1 x_1" seems to be an anomaly while "c_2 y_1" does not. To differentiate between these two scenarios, we use entropy, a measure of how predictable a distribution is. Specifically, we compute the entropy of each command,

$$H(c) = -\sum_{\langle a_i \rangle \in \mathcal{A}} P(\langle a_i \rangle \mid c) \log P(\langle a_i \rangle \mid c),$$

where $\mathcal{A}$ is the set of possible sequences of attributes for this command and $\langle a_i \rangle = (a_1, a_2, \ldots, a_n)$ is a particular sequence. In our example, c_1 has low entropy while c_2 has high entropy, thus a threshold weighted by entropy will differentiate between the two cases.

If the conditional probability of this line is significantly below the inverse of the entropy, the algorithm classifies the line as an anomaly. Specifically, the algorithm makes the following comparison

$$P(L_i \mid c) < \frac{\alpha}{H(c)}$$

where L_i is the i-th line and α is an empirically determined multiplier.

3. EVALUATION

We obtained 20 Cisco IOS router configuration files from the Carnegie Mellon campus network. A set of potential misconfigurations was found in the configuration files through manual inspection. We focused on those misconfigurations that the algorithm was designed to detect, specifically, the unusual or "lone" command.

If a command appears five or more times and, in all but one occurrence, takes one set of attributes, the unique occurrence is called a lone command. We found three lone commands in the Carnegie Mellon router configuration files. To confirm that these lone commands would be of interest to a network administrator, an expert familiar with the campus network reviewed them.

The sensitivity of our algorithms in detecting lone commands was determined by counting the number of false positives detected along with each misconfiguration. False positives are defined to be lines that are not lone commands. The detector was run on all 24 router configuration files. For each line, the minimum value of the detector parameter α necessary to classify the line as an anomaly was computed. For each lone command, we count the number of false positives with the same or lower minimum α value.

4. RESULTS

Figure 1 depicts the number of anomalies detected by joint Bayes as a function of the threshold multiplier (α). The minimum α value needed to detect each lone command is plotted with annotation on the curve. Joint Bayes detects two lone commands (Lone 2 and Lone 3) with no false positives. The other lone command (Lone 1) is only detected along with 2541 (out of a possible 11,125) false positives.

5. DISCUSSION

Our results show that joint Bayes is able to detect potential misconfigurations without also detecting other anomalies. Lone commands 2 and 3 are immediately detected by this detector. This type of misconfiguration is specifically that described by Caldwell et al. [1] as important to detect.

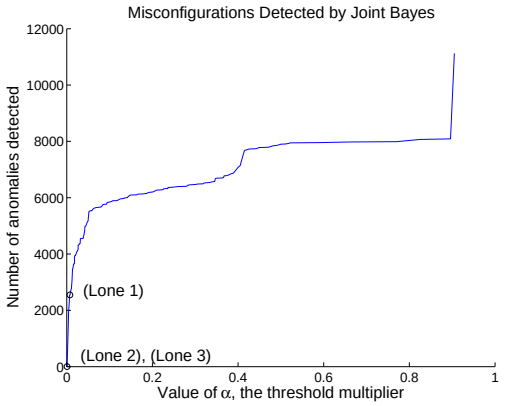

Figure 1: A comparison of the threshold multiplier α to the number of anomalies detected by the detector. The misconfigurations are plotted on the curve.

Further progress in detecting lone commands and other types of router misconfigurations as anomalies could be made if the assumption of independence between commands is relaxed. For instance, a misconfiguration in which interfaces are assigned undefined access lists could be detected if the dependency between the attributes of the `access-group` and `access-list` commands were taken into account.

6. CONCLUSION

The goal of this work was to determine whether router misconfigurations could be detected without prior knowledge of their form. A detector was designed and evaluated in this task, and it was able to successfully detect a certain type of misconfiguration in real-world router data.

7. ACKNOWLEDGMENTS

The authors are grateful to David Maltz for his significant assistance. This work was partially supported through National Science Foundation contract number CNS-0430474.

8. REFERENCES

[1] Don Caldwell, Anna Gilbert, Joel Gottlieb, Albert Greenberg, Gisli Hjalmtysson, and Jennifer Rexford. The cutting EDGE of IP router configuration. *ACM SIGCOMM Computer Communications Review*, 34(1):21–26, 2003.

[2] Nick Feamster and Hari Balakrishnan. Detecting BGP configuration faults with static analysis. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, pages 43–56. USENIX Association, Berkeley, CA, 2005.

[3] Anja Feldmann and Jennifer Rexford. IP network configuration for intradomain traffic engineering. *IEEE Network Magazine*, pages 46–57, September/October 2001.

[4] Declan McCullagh. How, why Microsoft went down. *Wired News*, January 25, 2001. `http://www.wired.com/news/technology/0,1282,41412,00.html`.