Minerals: Using Data Mining to Detect Router Misconfigurations

Franck Le, Sihyung Lee, Tina Wong, Hyong S. Kim, Darrell Newcomb

May 23, 2006
CMU-CyLab-06-008

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Minerals: Using Data Mining to Detect Router Misconfigurations

Franck Le Sihyung Lee Tina Wong Hyong S. Kim
ECE and CyLab
Carnegie Mellon University

Darrell Newcomb
Network Operations
CENIC

Abstract

Recent studies have shown that router misconfigurations are pervasive and have dramatic consequences for the operations of networks. Not only can misconfigurations compromise the security of a single network, they can even cause global disruptions in Internet connectivity. Several solutions have been proposed that can detect a number of problems in real configuration files. However, these solutions share a common limitation: they are rule-based. Rules are assumed to be known beforehand, and violations of these rules are deemed misconfigurations. As policies typically differ among networks, rule-based approaches are limited in the scope of mistakes they can detect. In this paper, we address the problem of router misconfigurations using data mining. We apply association rules mining to the configuration files of routers across an administrative domain to discover local, network-specific policies. Deviations from these local policies are potential misconfigurations. We have evaluated our scheme on configuration files from a large state-wide network provider, a large university campus and a high-performance research network, and found promising results. We discovered a number of errors that were confirmed and later corrected by the network engineers. These errors would have been difficult to detect with current rule-based approaches.

1 Introduction

Configuring routers is a tedious, error-prone and complex task. [11] presents a quantitative study of configuration errors in 37 firewall engines, and found that all of the firewalls contained some misconfigurations. The author's conclusion is "complex rule sets are apparently too difficult for administrators to manage effectively." [4] found more than 1000 errors in the router configurations of 17 networks while focusing only one aspect of the configurations – BGP. [7] measured BGP configuration errors that were visible from routing updates at the

Oregon RouteViews server over the course of 21 days, and found that misconfigurations were pervasive. For example, about 75% of all new routes advertised were erroneously announced during that time, a conservative estimate according to the authors. Although these misconfigurations did not disrupt Internet connectivity *per se*, they do contribute to router load and increase in routing convergence time. Other misconfigurations can have had a larger impact: in December 2004, misconfigurations in network AS9121 resulted in the propagation of 100K+ routes, leading to "misdirected/lost traffic for tens of thousands of networks" [1].

Several solutions have been proposed to deal with the router misconfiguration problem [5, 2, 4, 10, 3]. All but one of them compare configurations with a list of constraints or common best practices that a network ought to follow to function correctly. This approach of rule-based analysis makes the assumptions that rules violations are misconfigurations, and is very effective in detecting certain type of clear-cut problems. For example, ensuring all routers in a network are up-to-date on security patches [10], checking internal BGP speakers form a full mesh [4], and determining whether referenced routing policies are actually defined [5].

However, what constitutes an error sometimes depends on the network – what is an error for one network can be common practice for another. This relativism of error definition is echoed by others. [8] studied configuration files from 31 networks, and concluded that routing design can be diverse and each network is so different from another that it is not possible to classify them: the design of network routing is eclectic, "an art where many approaches might be used to try to achieve the same result". While, for routing designs, classic textbooks generally define two architectures – the enterprise and backbone architectures – 2/3 of the analyzed networks "exhibited designs that were markedly different from textbook examples and from each other". To give another example, in interdomain routing, a neighbor is usually categorized as either a provider, peer, or customer. However, we know of arrangements in a fairly large provider network that treat a neighbor as both peer and customer. Rule-based analysis presents limitations in the scope of errors it can detect. Its rules would have to be the "lowest common denominator", ones that are universally applicable to all networks.

Solutions to detect router misconfigurations fall on a spectrum. On one extreme, we can use tools that apply best common practice rules to detect known misconfigurations. On the other extreme, there is pure data mining that ignores underlying structure of router configuration commands and domain specific knowledge. We choose to be somewhere in the middle of this spectrum of the two extremes. *Minerals* applies data mining on router configuration files across a network to infer local, network-specific policies and detect potential errors that deviate from the inferred policies.

We discuss related work in the next section. Section 3 describes our approach, Minerals. Section 4 presents our evaluation of Minerals on configuration files from three different networks. The results are promising. Finally, we conclude the paper with discussions and future work.

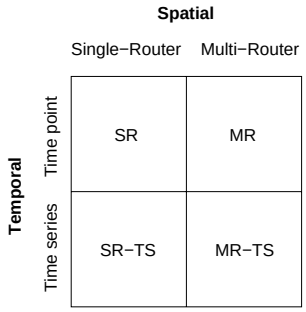

Figure 1: Taxonomy.

2 Related Work

In this section, we introduce a taxonomy to classify approaches that analyze network configurations and describe work related to Minerals in the context of this taxonomy. The taxonomy divides the approaches in two dimensions. The spatial dimension describes how many routers we study at a time. The temporal dimension represents whether we are looking at configurations from a specific point in time or across time. Figure 1 illustrates this taxonomy.

SR (Single-Router) explores one router's configuration from a specific time. RAT [10] checks a router's configuration file against security recommendations set forth by the NSA [9], highlights the differences and assigns a score to indicate the security level of the router in question.

MR (Multi-Router) studies multiple (or all) routers of a network from a specific time. Maltz et al. [8] developed a method to reverse engineer a network's routing design from its router configuration files. The main goal is to understand how operational networks use routing protocols. Xie et al. [12] extract information from configuration files to conduct static reachability analysis which can detect logic or design errors.

Feldmann and Rexford [5] are the first to propose misconfiguration detection through the parsing and analysis of router configuration files. rcc [4] aims to detect BGP misconfigurations by examining router configurations across a network. Some of the misconfigurations it can find include iBGP signaling problems and commands referring to undefined policies. Both [5, 4] are rule-based approaches in which rules or policies need to be known *a priori* and to be provided in advance. EDGE [2] argues that manual configuration of routers is error-prone and should be replaced by an automated process with inventory control. It suggests using data mining to mine configurations for errors but describes no details.

The work of El-Arini and Killourhy [3] is perhaps the closest to ours. They use a set of Bayesian based algorithms to detect statistical anomalies in router configurations. While the goal is similar, our approach is different. Their algo-

rithms focus on the frequencies of the command lines and associated attributes but do not consider the meaning of the commands. Instead, we preprocess the configuration files applying domain-specific knowledge to extract the meaning and relevant features. Also, the authors assume that each command line is independent and do not draw any relations between different commands. However, such assumption does not hold for router configurations as recognized by the authors.

SR-TS (Single-Router-Time-Series) looks at one router's configuration across time, and **MR-TS** (Multi-Router-Time-Series) analyzes multiple routers across time. EDGE suggests automating the provisioning tasks of a network by studying a network's configuration over time to identify recurring steps, but it provides no concrete method. To the best of our knowledge, there are no other proposals in the TS area.

3 Minerals

Data mining searches for patterns and rules in large data sets. Recently, it has been applied to systems and networking problems such as isolating bugs in software and detecting anomalies in traffic. Data mining can be used to identify errors in network configurations as well. Network elements often share common configurations. For example, to block spoofed packets, all edge routers in a network use filters blocking packets with invalid source IP addresses. Moreover, policies are usually applied across a network and evident in most routers in the network. Minerals uses data mining to discover local rules of a network and detects potential misconfigurations that deviate from these rules.

Local rules of a network can be complex and usually not captured by universal rules set forth by common best practice documents. To illustrate, we describe the usage of MD5 security to protect BGP sessions in a regional network provider and a large university campus. The "textbook" rule is to turn on MD5 if it is supported by the routers involved. Indeed, this is the local rule of the network provider. However, the stability of MD5 implementations varies across vendors and operating system versions. Some ASes resist turning on MD5 because it delays session re-establishment after a reset. The local rules in the case of the university network are MD5 is "off" if the end-point routers have exhibited problems in the past, "off" between certain ASes who preferred not to use it, "on" if the routers are from a particular vendor, and "on" on the rest of the sessions.

In this section, we first describe using association mining [6] to find patterns of correlation between elements in router configurations across a network – outliers to the discovered patterns are potential misconfigurations. The underlying assumptions are: there exists common configurations across routers, and the number of properly configured functions is large when compared to the number of their misconfigured counterparts. The obvious drawback of these assumptions is that non-conforming configurations can be classified as errors. To handle this drawback, we discuss incorporating time-series data – that is, historical con-

figurations of a network over time – to train and to improve accuracy of our technique. Note that logic or design errors which violate general network goals, such as incorrect packet filters preventing subnets from communicating, cannot be detected by Minerals.

3.1 Preprocessing the Input Data

In data mining, preprocessing the input data is essential to remove irrelevant information that can confuse a learning algorithm and yield useless results [6]. Experiments have shown that data mining is sensitive to the input data, and there is no exception here. Below, we describe how we preprocess router configurations using domain-specific knowledge before mining them. In our implementation, the preprocessing step is automated and is general so that almost all of it can be applied on different networks without modification.

3.1.1 Instance Type Identification

An *instance* is a unit of independent input data and association mining works on a set of instances. In Minerals, an instance is also the unit of error detection. The content of a configuration file can be divided into different *instance types*, in which each type is independent from the rest and undergoes separate analysis. The advantage of dividing into these smaller subsets is that we can reduce the amount of meaningless patterns discovered by association mining. The disadvantage is that we might miss patterns that exist across instance types but are not obvious to us.

In this paper, we focus on three instance types: user account, interface and BGP session. Many other instance types can be defined and mined for misconfigurations, such as route import policy, route export policy, packet filter, service, authentication scheme, VLAN definition, and so on.

3.1.2 Attribute Selection

An instance type is represented by a list of *attributes*. Not all attributes are important and suitable for data mining. Some attributes do not contribute to the operation of a network and should not be included (e.g., description strings). Other attributes are tweaked periodically for testing purposes or on a case-by-case basis and are not conducive to association.

We select attributes for each of the three instance types so that only relevant information is retained. User account is identified by the (router name, user name) pair and is also characterized by the attributes password, assigned privilege level and authentication method. BGP session is identified by the (router name, neighbor router) pair. We select the following attributes which characterize a BGP session at a high-level: the AS number of the neighbor router, incoming routing policies, outgoing routing policies and authentication scheme. For the interface instance type, there are a myriad of configuration options associated with it. We avoid attributes that are functionally indepen-

dent thus not conducive to data mining. We select an interface's IP address and whether it serves as a loopback.

3.1.3 Attribute Abstraction

To remove noise that might affect association mining, we further process the selected attributes as follows:

- *Boolean.* If an attribute is expected to have different values in different instances, it is not useful to compare the values themselves, but rather if the attribute exists. If we apply data mining on the attribute as is and compare the values directly, there would be few or no common patterns among them. Thus, in this case, we change the attribute so it takes on boolean values. For example, the value of the password attribute in the account instance type is not as important as the existence of an explicit, non-default password.

- *Group.* If an attribute has values that belong to groups and the values themselves differ in different instances, we convert the attribute so it takes on the group names as its values. The reasoning is the same as above. For example, instead of storing the actual IP addresses of the different interfaces, the IP addresses are classified as private or public. Similarly, the AS number is not employed alone but used to derive the type of BGP session (i.e., internal or external).

3.1.4 Data Cleaning

Raw configuration files or command lines are not conducive to data mining, especially when mining across a network with many routers. Each router vendor has its independent configuration language, e.g. Cisco's IOS vs Juniper JUNOS, that can be vastly different from others. An intermediate representation is necessary to compare configurations from different vendors, which may bear no resemblance syntactically but are semantically equivalent. To illustrate, IOS relies on privilege levels (0-15) to define a user's rights whereas JUNOS uses the concept of classes (superuser, operator, etc). An IOS's privilege level 15 can be considered comparable to JUNOS's "superuser" class. Sometimes an intermediate representation cannot save the day. Vendors can implement a feature in different ways and they cannot be reconciled across vendors. For example, there seems to be no equivalent of IOS's privilege level 6 in JUNOS. We have implemented parsers and defined appropriate intermediate representations for the three chosen instance types.

There can also be various ways to implement the same functionality. For example, in IOS, a routing prefix can be filtered on a BGP session either directly through applying a `prefix-list`, or by using `prefix-list` in a `route-map` and applying the route-map on the BGP session. We can clean the data such that attributes that define the same functionality are grouped into one attribute. Using the previous example, the attributes "incoming prefix-list" and "incoming route-map" can be subsumed into "incoming policies" as they both specify

Instance Type	Attribute	Value
Account	user name	string
	password	bool
	privilege	int
BGP session	type	{internal,external}
	MD5	bool
	incoming policies	bool
	outgoing policies	bool
	AS number	int
Interface	loopback	bool
	IP addr	bool
	IP addr type	{private,public}

Table 1: Summary of preprocessing outcome, which serves as input to association mining algorithm.

routing policies.

The results of the preprocessing for each of the three instance types are summarized in Table 1. They serve as input to the association mining algorithm, which is described next.

3.2 Association Rules Mining

We give a brief overview of association rules mining; interested readers can refer to [6] for details. At a high level, association rules mining examines the statistical properties of correlations present in a large data set. In the traditional data mining context, it is used to classify and predict an attribute or combination of attributes. The prediction is called a *rule*. Let $a, b, c, \ldots$ be attributes. Association mining searches for rules $((a = a_i \wedge b = b_j \wedge \ldots) \rightarrow m = m_k)$, in which any combination of unique attributes can be on the left-hand side. In other words, a rule takes the form "if X then Y", or $X \rightarrow Y$, where the left-hand side represents the "condition" and the right-hand side the "consequent". The pattern $(a = a_i \wedge b = b_j \wedge \ldots)$ is also known as an *itemset* – $(a = a_i)$ is a 1-item itemset, $(a = a_i \wedge b = b_j)$ a 2-item itemset, and so on. Given an itemset X, its *support*, $S(X)$, is defined as the number of instances that satisfy the condition X. Given $X \rightarrow Y$, its *confidence*, $C(X \rightarrow Y)$ is calculated as the $S(X \wedge Y)/S(X)$. We illustrate association mining using Table 2: $S(a = a_1) = 3$, $S(a = a_2) = 2$, $S(a = a_1 \ \wedge \ b = b_1 \ \wedge \ c = c_1) = 2$; $C(a = a_1 \rightarrow b = b_1) = \frac{3}{3} = 1$, $C(a = a_1 \rightarrow c = c_1) = \frac{2}{3} = 0.67$, and $C((a = a_1 \ \wedge \ b = b_1) \rightarrow c = c_1) = \frac{2}{3} = 0.67$.

In our context, rules with high confidence are considered reflections of local network policies. Instances that deviate from these rules are identified as potential misconfigurations because they do not comply with the inferred policies. We believe association mining is well suited for detecting anomalies in router configurations because network policies usually present the following properties:

- Network policies can sometimes be written as correlations, if not causal

Instance ID	a	b	c	...
1	a_1	b_1	c_1	...
2	a_1	b_1	c_1	...
3	a_1	b_1	c_2	...
4	a_2	b_2	c_1	...
5	a_2	b_2	c_2	...

Table 2: Illustration of association rules mining.

relations. For example, the policy "BGP sessions with external ASes must be protected" can be expressed in the rule (eBGP $= 1 \rightarrow$ MD5 $= 1$).

- To be effective, network policies are usually applied on most or all objects of the same category across a network. For example, ingress filtering to prevent spoofed packets should be applied on all border routers, not just one, and this policy can be as (external interface $= 1 \rightarrow$ bogonfilter $= 1$).
- Deviations from local network policies can either be misconfigurations, non-conforming valid configurations or "hacks", all of which should be audited periodically.

In Minerals, the generation of association rules and detection of violations from those rules consist of four steps:

Step 1: Itemset generation. For each instance type, this step generates all possible combinations of itemsets, e.g. all 1-item itemsets, 2-item itemsets, etc, with the restriction that the support is over the threshold `min_supp`. The default `min_supp` is 10. A number of algorithms have been proposed by the data mining research community to compute itemsets efficiently.

Step 2: Inference of local policies. The goal of this step is to generate association rules. Minerals filters out rules with confidence values lower than the threshold `min_conf` as we want to find rules that are the most pertinent and more likely to be reflections of local policies. The default `min_conf` is 0.9. This step can also apply domain-specific knowledge to eliminate irrelevant rules thus reduce the false alarm rate: if common sense says attribute a cannot imply or be correlated with b, we eliminate all rules of the form $\ldots \wedge (a = *) \wedge \ldots \rightarrow b = *$. Even though this only needs to be done once and can be henceforth remembered by the Minerals algorithm, it nevertheless can be labor intensive. We show in our evaluation that without applying this rule filtering we are still able to achieve decent detection rates.

Step 3: Violation detection. Since we are focusing on misconfigurations, we do not consider rules with confidence equal to 1. Violations are instances that do not comply to a rule $X \rightarrow Y$ with `min_conf` $< C(X \rightarrow Y) < 1$. The attribute that might be misconfigured is in Y – the rule has the expected, common value, whereas the violation has a potentially wrong value. We also eliminate rules that generate more than `max_violations` number of violations.

The reason behind using `max_violations` is the number of misconfigurations should be small in a network. If a rule results in a large number of violations, the violations are most probably not misconfigurations, and the rule unlikely to reflect a local policy. `max_violations` can be proportional to total number of instances of the instance type in question. We use a simple default value of 10.

Finally, we can remove violations of a rule $((a = a_i \wedge \ldots \wedge b = b_j) \rightarrow m = m_k)$ if the violations comply with a longer rule $((a = a_i \wedge \ldots \wedge b = b_j \wedge \ldots) \rightarrow m \neq m_k)$. For example, instances that violate a rule (eBGP $= 1 \rightarrow$ MD5 $= 1$) may conform to another rule (eBGP $= 1 \wedge$ AS number $= 9 \rightarrow$ MD5 $= 0$) with an additional attribute, AS number. This can imply that the local rules of AS 9 prefer not to use MD5 passwords. By eliminating such violations, the tradeoff is we can reduce the number of false alarms but might miss some real errors.

Step 4: Filter and report. The last step reports the identified violations in decreasing order of confidence. Since multiple rules can point to the same misconfigured instance, we only report the instance once and indicate the rule with the smallest item set size on the left hand side.

3.3 Time Series Analysis

The time dimension can be utilized to train a data mining algorithm in order to increase its accuracy. We can apply Minerals periodically or every time a router's configuration is modified. Setting the thresholds in Minerals is a tradeoff between number of false alarms and number of detected errors. One can slowly increase `min_supp` and `min_conf` and slowly decrease `max_violations` over time, so that the mining algorithm can be trained with more data in the beginning.

After each run of Minerals, the network operator can provide direct feedback on whether and how a reported violation is a false alarm: the operator can indicate that the mined rule is not a reflection of his network policy, or the violation is an exception to a valid policy. This knowledge can then be fed back into subsequent runs of Minerals to filter rules and violations. Alternatively, to ease burden from the operator, Minerals can learn indirectly. Violations that are reported but not corrected over time are assumed to be exceptions.

Also, analysis of the evolution of a network's configuration over time can reveal additional misconfigurations not detected by snapshot analysis. We can data mine successive snapshots of the configuration to expose patterns of change. We took monthly snapshots of the CalREN-DC and CalREN-HPR configurations from June 2004 through March 2006. We analyzed the evolution of user accounts on the routers which revealed some interesting patterns. A handful of accounts are rotated regularly: a username would be added to almost all routers at about the same time, then deleted some time later, and replaced by another username. This turns out to be backdoor accounts that are created to ensure management access during times of failure, DoS or planned maintenance events. Data mining can be applied to discover outliers in this pattern. E.g. routers that are misconfigured during rotation which either have multiple backdoor accounts, or new routers that are overlooked and have no backdoor accounts. We

Network name	Num. of routers total (IOS,JUNOS)	Num. of LOC (min,max)
CalREN-DC	44 (37,7)	(230,3060)
CalREN-HPR	6 (6,0)	(455,3638)
UC Berkeley	67 (65,2)	(92,1688)

Table 3: Summary of networks used in evaluation. CENIC is the California state-wide network service provider for education and research institutions: CalREN-DC is a backbone network serving education users, and CalREN-HPR is a high-performance research network providing advanced services for large application users. LOC stands for lines of commands.

also analyzed the use of private IP addresses over time, and observed that many interfaces were starting to use them at about the same time, and released some time later. Private IP addresses are sometimes used for testing. If some interfaces with private IP addresses are not removed while the rest are, after testing is finished, a time series data mining algorithm can pick out these anomalies.

4 Evaluation and Results

We implemented and evaluated Minerals on the configuration files from three networks. In this paper, for each of the three network, we analyzed a particular snapshot of its configuration files, taken between January and March 2006. Information about the networks is summarized in Table 3. We used default values of `min_supp`, `min_conf` and `max_violations`. We did not pre-filter any association rules to allow the algorithm to find all possible policies and violations.

Our results show that Minerals is applicable to networks of various sizes. Although CalREN-HPR has only 6 routers, Minerals still detected several misconfigurations within it. Minerals detected misconfigurations in each network, and these configurations were confirmed by the operators. While some of the errors can be classified as benign, others were more severe and were able to expose the entire network to attackers. Many of the errors Minerals detected are violations of local network policies, which approaches based on universal rules cannot detect.

Table 4 summarizes the results we obtained on the three instance types: account, BGP and interface. We also included missing definition errors found by our parser. These are definitions that are referred to but not defined anywhere in a router's configurations. Rule-based approaches can spot these errors as well. We divide violations flagged by Minerals into three categories. A confirmed error is confirmed by the network operator as an error, and usually fixed afterward. A false alarm is a non-conforming but valid configuration. A "Needs investigation" is when the reason for a particular flagged configuration was not immediately clear to the network operator and required further investigation.

Unused interfaces. In one network, 95% of the interfaces had public IP

Network name	Definition	Account	BGP	Interface
CalREN-DC	(2,0,0)	(9,6,1)	(2,0,0)	(4,0,0)
CalREN-HPR	(6,0,0)	(0,0,0)	(3,2,0)	(6,0,0)
UC Berkeley	(1,0,0)	(0,0,0)	(0,4,1)	(0,0,0)

Table 4: Summary of quantitative results of Minerals. The numbers in parenthesis represent (confirmed error, false alarm, needs investigation).

addresses, and Minerals highlighted interfaces using private IP addresses as violations. According to the operator, private IP addresses are used to bootstrap routers at the beginning of deployment, and should be deleted afterwards. These confirmed misconfigurations were corrected by the operator. It is difficult for rule-based approaches that rely on universal rules to detect such errors, because the presence of private IP addresses does not automatically imply a misconfiguration. Some networks use private IP addresses permanently, e.g., for network management devices. The advantage of data mining is it considers the statistical properties of a network, and in this case, reveals the outlier interfaces as errors.

Missing passwords. In one network, Minerals found accounts without passwords, which was confirmed by the operator as errors. Some of these accounts had super-user privileges, which presented a security breach. Universal rules might not be able to capture different account policies in different networks. Some networks create accounts without passwords to allow access from external parties, e.g. to view routing tables. A network can assign super-user privilege to only one user, while permitting normal privilege for the rest of the users. Minerals can find these patterns.

Omitted export policies. In one network, eBGP sessions that applied import policies also had export policies, except two sessions. The operator confirmed that the two sessions were errors and described them as "very concerning". The absence of export policies can result in a list of undesired effects, such as unintentionally providing transit service.

Lacking MD5 security. In one network, Mineral detected three eBGP sessions were missing MD5 security, and they were confirmed as oversight errors by the operator. The lack of MD5 security on a eBGP session renders the router vulnerable to the TCP RESET attack.

False alarms and others. Minerals reported 12 violations that were false alarms – they were non-conforming but valid configurations. Two eBGP did not have MD5 security because the neighbor AS preferred not to turn it on. Four eBGP sessions did not present any export policy. An interesting set of false alarms involves 2 usernames u_x, u_y on 3 routers r_a, r_b, r_c, i.e. 6 accounts $(u_x, r_a), (u_y, r_a), (u_x, r_a), (u_y, r_b), (u_x, r_c), (u_y, r_c)$. Minerals found a local rule that says all accounts on the same router should have the same privilege level.

The 6 accounts were special case configurations for 2 neighbor ASes and were valid on the 3 routers.

5 Discussions: Challenges and Directions

We are exploring adding attributes other than the ones listed in Table 1 in Minerals. In particular, besides IP address and loopback, a variety of options can be enabled (or disabled) on an interface. A routing protocol can be associated with an interface as well. We have started to apply Minerals on interfaces with these additional attributes, and found a number of confirmed errors. In one network, a majority of interfaces had proxy-ARP disabled to prevent issues where large ARP caches can develop in poorly connected devices. Minerals found the 39 interfaces which violate this rule. Another confirmed misconfiguration Minerals found is two eBGP interfaces with Cisco Discovery Protocol (CDP) enabled when the common pattern in that network is to have CDP disabled. CDP can expose details about a network to external parties which is often undesirable. However, there are a number of challenges when using additional attributes in Minerals. Some of the attributes may take actions that differ from their specified values. As an example, IP Redirect is ineffective on loopbacks since packets do not arrive on these interfaces. Some of these attributes can be functionally independent, thus not conducive to data mining. We are investigating the meaning behind various interface options, and how networks tend to configure them. We believe this domain specific knowledge can be applied to Minerals to address these issues.

We are also encountering a number of challenges with respect to configuration commands in this project. First, it was at times difficult to equate features supported by IOS and JUNOS. Second, the default values for some attributes are different between different versions of router operating systems. For example, in older versions of IOS, "directed broadcast" was enabled by default, whereas in more recent versions it is disabled by default. Thus, the absence of "[no] ip directed broadcast" command has entirely different meanings depending on the IOS version. This disparity complicates the preprocessing of input data. We believe a new abstraction for network configuration management would be ideal, one that remove these details and allow network operators to work at a higher level and across different vendors. Despite these obstacles, Minerals was successful in detecting a number of errors that would have otherwise flown under the radar of other techniques.

Acknowledgments

This work was funded in part by KISA, MIC, Samsung, ARO and Carnegie Mellon CyLab. Imran Suilaiman and Shawn Kim contributed to the implementation of the configuration file parser. We would like to express our gratitude to Ken Lindahl, George Kaplan and others at CNS for giving us access to Berkeley's configuration files, for comments and suggestions, and for educating us

about network operations. Tina wants to thank Michael Sinatra for adding humor to our project by betting that Berkeley's (mis)configurations could cause Minerals to blow up like the "Robert rolled reddish radishes around the ring of rubber rubbish" machine in Seasame Street.

References

[1] B. J. P. Alin C. Popescu and T. Underwood. Anatomy of a leak: As9121 (or, "how we learned to start worrying and hate maximum prefix limits"). In *NANOG 34*, May 2005.

[2] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford. The cutting edge of ip router configuration. In *Proceedings of HotNets-II*, Boston, MA, November 2003.

[3] K. El-Arini and K. Killourhy. Bayesian detection of router configuration anomalies. In *Sigcomm Workshop on Mining Network Data*, 2005.

[4] N. Feamster and H. Balakrishnan. Detecting bgp configuration faults with static analysis. In *Proceedings of NSDI*, Boston, MA, May 2005.

[5] A. Feldmann and J. Rexford. Ip network configuration for intradomain traffic engineering. *IEEE Network Magazine*, 2001.

[6] E. F. Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[7] R. Mahajan, D. Wetherall, and T. Anderson. Understanding bgp misconfiguration. In *Proceedings of Sigcomm*, Pittsburgh, PA, August 2002.

[8] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg. Routing design in operational networks: A look from the inside. In *Proceedings of Sigcomm*, Portland, OR, September 2004.

[9] Router security configuration guide. System and Network Attack Center, National Security Agency, December 2003. Available at http://www.nsa.gov/snac/routers/cisco_scg-1.1b.pdf.

[10] The router audit tool (rat). http://www.cisecurity.org/bench_cisco.html.

[11] A. Wool. A quantitative study of firewall configuration errors. *IEEE Computer*, June 2004.

[12] G. G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford. On static reachability analysis of ip networks. In *Proceedings of Infocom*, 2005.