Characterization and Problem Detection of Routing Policy Configurations

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

June 12, 2006
CMU-CyLab-06-010

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Characterization and Problem Detection of Routing Policy Configurations

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

June 13, 2006

Abstract

The current Internet routing landscape presents a number of challenges, especially in the configuration of routing policies. There have been numerous proposals to tackle the misconfiguration problem: configuration checking, policy language (re)design, and clean-slate routing architecture. In this paper, we present an analysis of routing policies and a misconfiguration detection mechanism. With an operational perspective, we first present a study on the configuration and evolution of routing policies, using data from three different types of networks. Our results show that configurations are changed frequently and mostly incrementally. We found that the most commonly used and changed commands are related to route tagging and filtering, and there are substantial amount of duplication in policy configurations within a network. More interestingly, based on these results, we develop a data mining method to find inconsistencies in a network's configurations of routing policies. Our method is able to detect local, network-specific rules automatically, and differs from existing approaches that are based on universally applicable rules. In our evaluation, we found 30 confirmed errors and 29 warnings in three networks. More than half of the errors are related to route tagging. Our findings show that the next generation configuration language and routing platform should be sufficiently flexible to allow a network to express and frequently modify its route tagging, yet restrictive enough as this aspect is often misconfigured.

1 Introduction

Many have compared configuring a network to writing a distributed program in assembly language. In particular, the configuration of routing protocols and their associated policies is a complex task reserved for highly experienced network operators. Routing policies can be configured to achieve traffic engineering

*Darrell Newcomb is with Network Operations at CENIC. The rest of the authors are affiliated with the Department of ECE and CyLab at Carnegie Mellon University.

within a network, maintain appropriate business relationships with neighbor domains, and defend the network from outside attacks. If configured improperly, routing policies can cause many undesired outcomes both in the internal operation of the network and also externally in the Internet.

Usually, configuring a network is performed manually, by logging into and manipulating each router separately. There are also commercial products in configuration management which help to track changes in the configurations and keep router software up-to-date. However, most of these products only work in networks with devices from a single vendor, which is not always a practical requirement. A number of studies have shown that misconfigurations are not uncommon in networks big and small, especially with respect to interdomain routing policies [16, 21, 9].

Several researchers have proposed solutions to tackle the misconfiguration problem [10, 7, 9, 20]. These solutions compare router configuration files with a set of "best common practice" rules that all networks ought to follow. While this rule-based approach is very effective in detecting certain types of errors, it relies on rules that are the "lowest common denominator" to all networks. Thus, the scope of errors the rule-base approach can detect is limited. Unless an operator works with the tool developer to define rules for the network on a case-by-case basis, the peculiarities of the network's configurations would not be considered in the error detection process.

Other researchers have taken the approach of re-designing configuration languages [3, 13]. One goal is to provide a vendor-neutral language with high level constructs to represent routing policy in a more abstract manner than the current vendor-specific primitives. Another goal is to embed constraints in the language such that the resulting policy would have certain consistency guarantees. Designing a language that would allow global policy coordination among autonomous domains or even local coordination within a network is a challenging problem.

Although one would like to blame router vendors for all (mis)configuration problems, the current state of the matter has culminated from an "organic" evolution of the Internet [13]. Recently, there have been several proposals of clean-slate routing architectures, which advocate the separation of routing policies from routers [12, 5]. To support this direction, Maltz et al. conducted the seminal work of reverse engineering a network's routing design from its router configuration files. Their results provide insights on how to structure the next-generation routing architecture.

All three problems described above – configuration checking, policy language design and new routing architecture – remain open research problems [22, 6]. In this paper, we aim to contribute to solving these problems in two ways. First, we provide empirical observations that would be useful in tackling these problems. With an operational perspective, we analyze the configuration and evolution of routing policies in several different networks. To the best of our knowledge, this is the first paper to present a longitudinal study of router configuration files; while there are several previous studies on configurations, single snapshots are used. More importantly, based on these empirical observations, we have de-

signed and developed a novel way to detect misconfigurations in routing policies. Our method is based on data mining and is different from existing rule-based approaches.

Our contributions are summarized as follows:

- Using router configuration files from three different networks – a large university campus, a statewide transit network provider, and a research network – we characterize the configurations of routing policies over periods of 3 to 21 months, depending on the network (§4). Our analysis show that configuration changes are made frequently and incrementally. We found that the top 10 most common commands used to configure routing have to do with route tagging and filtering (§4.1.1). For one of the networks, routing related changes account for 55% of all changes made over 21 months. (§4.1.2).

- We present two new metrics, risk and impact, to measure the complexity of route tagging in a network. In two networks we studied, we found that route tagging exhibit complex usage behavior, and its risk and impact tend to grow over time (§4.2). We also present a metric to assess the duplication factor of routing policies across multiple routers within a network. We found that the duplication factor is high. One network presents 407 patterns of routing policies in which each pattern is present in at least two BGP sessions (§4.3).

- §5 contains the major contribution in this paper. Based on the above observations, we have developed a data mining method using an association rule mining algorithm, called Minerals, to detect misconfigurations in routing policies (§5.1). We applied Minerals on configuration files from the three target networks. Our results show that Minerals is able to detect local, network-specific rules and find inconsistent configurations that deviate from these rules. It found 30 confirmed errors, 29 warnings, and 16 false alarms. More than 50% of confirmed errors are related to route tagging. We also found that configuration changes are almost always applied in gradual steps, seldom all-at-once. (§5.2)

Caveats: Configuration files are often considered sensitive data and kept confidential. Because of the difficulty in collecting configuration files from operational networks, our study presents a number of limitations. First, our study examines the configuration files from a limited number of networks. Second, only three architecture types are included. Our dataset does not contain configurations from commercial networks nor Tier-1 providers. Third, the history of the configurations for one network spans over the period of only three months. Despite these limitations, we believe the observations and results are still meaningful. The proposed methods are general and still applicable to other networks. Finally, the misconfiguration detection scheme can help operators of existing networks in detecting configuration errors.

```
1   router rip
2    passive-interface Ethernet0/0
3    network 1.2.3.0
4    distribute-list 1 out Ethernet0/0
5
6   router bgp 100
7    neighbor dora peer-group
8    neighbor dora remote-as 200
9    neighbor 4.5.6.1 peer-group dora
10   neighbor dora prefix-list pf_dora in
11   neighbor dora route-map from_dora in
12   neighbor dora route-map to_dora out
13
14  ip community-list 2 permit 100:4
15
16  ip prefix-list pf_dora permit 10.10.0.0/16
17  ip prefix-list pf_dora permit 10.11.0.0/16
18
19  access-list 1 permit 0.0.0.0
20  access-list 1 deny   any
21
22  route-map from_dora permit 100
23   set local-preference 100
24   set community 100:1 100:2 100:3
25
26  route-map to_dora deny 10
27   match community 2
```

Figure 1: Excerpt of a Cisco router configuration file.

2 Background

In this section, we briefly describe the configuration on routing protocols and policies. Readers with experience in this area can skip to §3.

Routing policies control which routes a router or a network accepts, filters and forwards. At a high level, routing policies allow a network to specify the flow of its inbound and outbound traffic. Redistribution of routes between routing processes (e.g. from OSPF to BGP, from RIP to OSPF) can also be configured. Note that routing policies are not carried in routing protocol messages such as BGP UPDATE messages. Routing policies are private to a network, and are specified on the routers that collectively form a network.

Each router vendor has its own proprietary configuration language. Some configuration features are vendor-specific and not available in another vendor. In the rest of this paper, we use Cisco IOS terminology and syntax, but both our methodology and implementation work for Cisco IOS and Juniper JUNOS. Figure 1 is an factitious excerpt of a router configuration file. We use it to illustrate how routing protocols and policies are defined.

Lines 1-4 configure the RIP routing process. The `passive-interface` command prevents routing updates from being sent on the specified interface `Ethernet0/0`. The `network` command says all interfaces connected to `1.2.3.0` are to use RIP.

Network name	Timespan	Num. of routers total (IOS,JUNOS)	Num. of LOC (min,max)
DC	Jun 04-Mar 06	44 (37,7)	(230,3060)
HPR	Jun 04-Mar 06	6 (6,0)	(455,3638)
UCB	Mar 06-May 06	67 (65,2)	(92,1688)

Table 1: Summary of networks used in evaluation. LOC stands for lines of commands.

`distribute-list out` restricts advertised updates according to the access list called `1`, defined in lines 19-20. The access list permits a default route to be advertised.

Lines 6-12 configure the BGP routing process. The number `100` is the local autonomous system (AS) number. In line 7, the command `peer-group` creates a group called `dora`. `peer-group` facilitates the application and modification of routing policies on a set of neighbors. Line 8 associates the AS number `200` to `dora`, which tells us whether the BGP session is with external or internal neighbors. Line 9 assigns the BGP neighbor with IP address `4.5.6.1` to the peer group `dora`. Line 10 applies a prefix list called `pf_dora` to all BGP sessions of `dora`. `pf_dora` is defined in lines 16-17, and permitting only the prefixes `10.10.0.0/16` and `10.11.0.0/16`. The keyword `in` specifies the prefix list is applied to incoming advertisements. Therefore, only `10.10.0.0/16` and `10.11.0.0/16` are accepted from the peer group `dora`.

Lines 11-12 apply the import policy `from_dora` and export policy `to_dora` to the BGP sessions in `dora`, which are defined in lines 22-24 and lines 26-27, respectively. `from_dora` accepts all routes and assigns them a local preference value of `100`. It also tags the routes with three communities `100:1`, `100:2` and `100:3`. `to_dora` denies routes which carry the community tag listed in the community list `2`, defined in line 14, which corresponds to the community `100:4`. Consequently, routes marked with `100:4` are not advertised to BGP neighbors of `dora`.

3 Datasets

Our dataset comprises of configuration files from the CENIC CalREN-DC and CalREN-HPR networks, and the UC Berkeley campus network. Information related to each network is summarized in Table 1. Two of the networks include both Juniper and Cisco routers and our study considered both types of routers. We collected the router configuration files from UCB over a period of 3 months and the ones from DC and HPR for 21 months. We do not include the hundreds of customer premise equipment (CPE) that CENIC manages in this paper, even though these periphery devices also contain policies.

Even though we analyzed only three networks, we believe that the dataset is representative of existing networks and covers different architecture types. DC is a regional transit AS serving a large base of universities and research

institutions. It peers with a number of commodity peers and buys services from multiple upper tier providers. HPR is a smaller research network providing advanced services for large application users. UCB is a stub network, i.e., an edge AS that buys service from a provider. Our data include the four types of relationships between autonomous domains [11]: customer-to-provider, provider-to-customer, peer-to-peer and sibling relationships. DC and HPR are siblings.

Because DC and HPR are non-profit organizations, their policies may not be as restrictive as a typical Tier 1 commercial provider. However, these two networks peer with hundreds of network to reduce their operational costs. Therefore, we believe that the observed patterns can still be meaningful, and the proposed methodologies can still be general and applicable to most networks.

4 Characterization of Router Configurations

This section examines the composition of configuration files with a focus on routing policies. §4.1 identifies the most commonly utilized and updated commands and attempts to determine the rationales behind the observed patterns. We found that a router's configuration can be modified rather frequently albeit only several lines at a time. Our analysis shows that commands related to the configurations of route tagging and filtering are among the most used and modified. Based on these results, §4.2 introduces a number of measures to assess the risk and impact in the configurations of BGP communities. Finally, §4.3 evaluates the degree of similarity among routing policies within a network. We found that a considerable proportion of routing policies in the studied networks show a high amount of duplication.

4.1 General Patterns

Router configuration files contain commands related to the operations of a router. They include information about the booting and startup, login, remote access, auditing, interfaces and routing protocols. When considering the number of lines related to the different parts, we found that the fraction related to routing policies can represent a predominant portion of configuration files. In large size files, the section related to the configuration of BGP routing policies can contribute to up to 66% of a file. Next, we identify and analyze the most prevalent and frequently modified commands in the configuration of routing policies.

4.1.1 Prevalent Commands

We determine the most frequently used commands by identifying vendor language keywords in each line of each router configuration file. We consider IOS and JUNOS separately because of the differences in their syntax. For both vendor language, we filter keywords and non-keywords that do not contribute to the

classification of commands. For example, line 10 (`neighbor dora prefix-list pf_dora in`) of Figure 1 is filtered to become `neighbor prefix-list`, which we count as a single command. We remove `dora` and `pf_dora` because they are user-defined names thus non-keywords, and the keyword `in` because including it would provide an overly fine-grained classification. The JUNOS language includes lexing structures such as `then` and `term` that we filter as well. We do not include booting and startup messages that are part of a configuration file in this analysis.

Table 2 lists the the top 10 most common commands used to configure routing policies for DC and UCB. The commands used in HPR are not represented since the results are very similar to DC. We do not illustrate the JUNOS commands in UCB because of space limitations. However, the analysis includes all three networks and both types of routers. It appears that:

- Commands related to route tagging (`community, community-list, community members, ip route tag`) are predominant in the observed networks.
- Commands related to route filtering (`route-filter, prefix-list, distribute-list`) are also frequently used.

These results may stem from the way certain routing policies are defined and how we count them. In IOS, to define a prefix list with 10 prefixes, 10 lines of `ip prefix-list` can be listed. Likewise, in JUNOS the command `route-filter` is used in a similar way. While our counting method could have overstated the role of the `ip prefix-list, ip community-list, route-filter`, we do not believe the dominance of route tagging and filtering is an artifact of our methodology. Other commands using these features, e.g. `match community, set community, set comm-list, match ip address prefix-list`, also show up in the top 10 tables.

In the DC and HPR networks, prefix lists are used to filter route advertisements from customers and commodity peers, and also to limit route readvertisements to commodity peers. CENIC also relies heavily on BGP communities to tag routes and control announcements [1]. Besides the usual control among provider, peer and customer relationships, there are several special arrangements with neighbor networks.

UCB shows a different usage pattern. This is in part due to the fact that UCB is a stub network: it does not provide transit and peers with few networks. Also, in UCB, the two Juniper routers are border routers connected to its service provider, and the Cisco routers are core routers running RIP and OSPF. The `distribute-list` command limits route redistribution in RIP, OSPF and BGP within UCB. The BGP related commands specify route preferences in order to implement traffic rate-limiting. BGP is also used for a blackhole routing system to protect against DoS attacks.

4.1.2 Prevalent Changes

Are router configurations modified frequently? Are most modifications small tweaks and big overhauls? Are the most common commands also changed often?

DC (IOS)		
Command	**Count**	**Fraction**
ip prefix-list	2852	0.14
ip community-list	2564	0.12
neighbor peer-group	1839	0.09
route-map	1637	0.08
neighbor password	1546	0.07
ip route tag	1227	0.06
match community	834	0.04
match ip address prefix-list	818	0.04
set community	795	0.04
neighbor route-map	530	0.03
		Cum. 0.72

DC (JUNOS)		
Command	**Count**	**Fraction**
community	612	0.15
community members	409	0.10
route-filter	344	0.08
neighbor	329	0.08
authentication-key	286	0.07
as-path	279	0.07
interface	117	0.03
policy-statement	116	0.03
level metric	112	0.03
next-hop	99	0.02
		Cum. 0.69

HPR (IOS)		
Command	**Count**	**Fraction**
ip prefix-list	1114	0.18
route-map	796	0.13
neighbor peer-group	538	0.09
match community	510	0.08
ip community-list	505	0.08
neighbor route-map	279	0.04
set community	256	0.04
set local-preference	206	0.03
neighbor activate	201	0.03
set comm-list	188	0.03
		Cum. 0.76

UCB (IOS)		
Command	**Count**	**Fraction**
distribute-list	650	0.18
neighbor peer-group	277	0.08
ip community-list	229	0.06
no passive-interface	215	0.06
network	202	0.06
ip route	182	0.05
route-map	155	0.04
ip prefix-list	115	0.03
passive-interface	114	0.03
neighbor remote-as	89	0.02
		Cum. 0.64

UCB (JUNOS)		
Command	**Count**	**Fraction**
route-filter	64	0.11
policy-statement	40	0.07
community	37	0.06
interface	34	0.06
community members	34	0.06
neighbor	27	0.05
prefix-list	26	0.05
group	25	0.04
export	25	0.04
type	23	0.04
		Cum. 0.62

Table 2: The top 10 most common commands used to configure routing policies, as observed in the configuration files of DC, HPR and UCB.

We attempt to answer these questions here.

In our first analysis, we study how configurations change in general in the three networks. For each router in a network, we look at how its configurations are modified between snapshots. For DC and HPR networks, we use daily

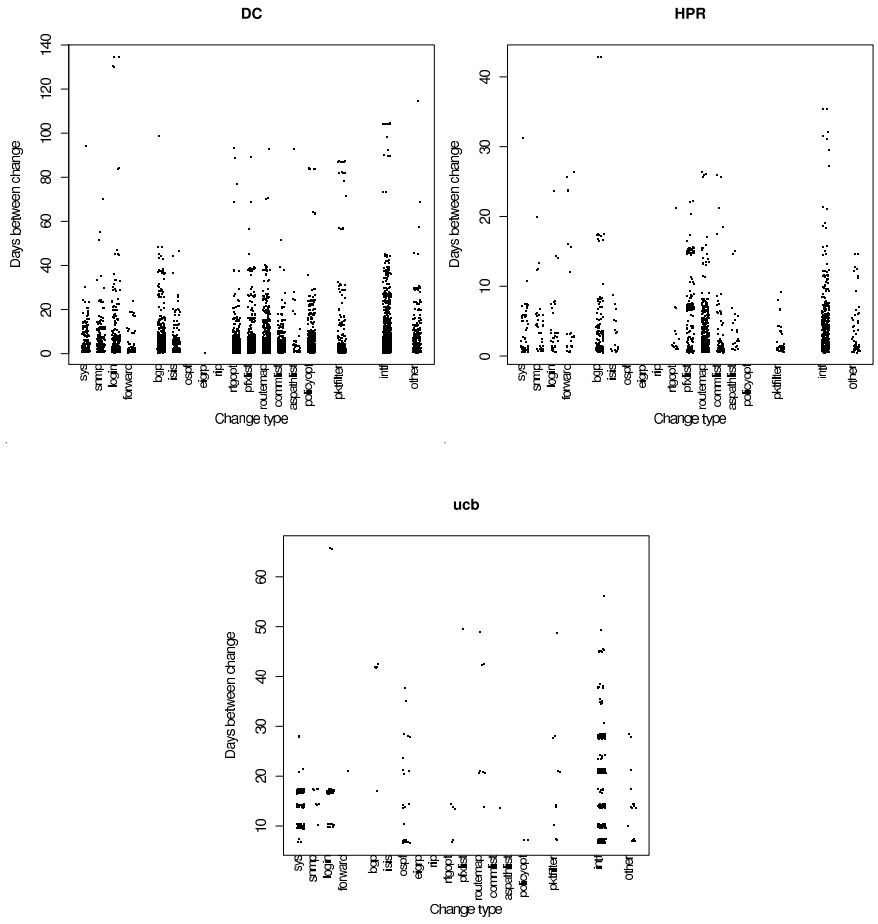


Figure 2: Types of changes on a router and their frequencies. The y-axis denotes the amount of time elapsed between change on a router, and the x-axis lists the categories of changes. Each dot in a graph represents a consecutive block of changes (additions, deletions or modifications) in a router. We add slight random jitter to each dot's position so it can be distinguished from other dots.

snapshots over 21 months, and for UCB, the snapshot granularity is between 7 to 14 days over 3 months. The results are illustrated in Figure 2. Each change is represented by a dot in a graph (changes to consecutive lines of a configuration file is counted as a single change). The y-axis shows the amount of time that has elapsed between changes to a router, and the x-axis indicates the category to which the change belongs.

A number of observations stand out from the graphs in Figure 2. Routing related modifications:

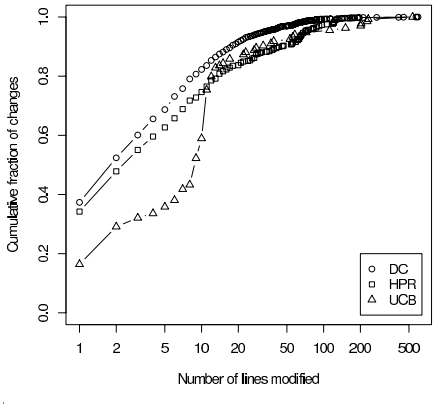


Figure 3: Distribution of number of lines modified during between snapshots: for DC and HPR, daily snapshots are used; for UCB, 7-14 day snapshots are used. The x-axis is log-scale to show details of modifications involving few number of lines.

- are made frequently, especially in DC and HPR, as illustrated by the dense clusters of dots towards low end of y-axis. Most of these changes occur within 10 days of the previous change to the same router.
- account for 55%, 54% and 7% of all changes made in DC, HPR and UCB, respectively.
- happen more often than changes to packet filters, even in UCB which is a stub network.

We also investigate the number of lines touched when a router's configuration is modified. Figure 3 illustrates the distribution of number of lines touched between our snapshots, for DC, HPR and UCB. Even though weekly and bi-weekly snapshots are used for UCB, 80% of modifications involve 15 lines or less. Single line changes account for 37%, 34% and 16% of all changes for the three networks, respectively.

In the second part of our analysis, we identify commands that appear the most frequently in changes. We found that:

- In all three networks, 80% of the changes related to routing policies are concentrated in 10 commands.
- In DC and HPR, the commands related to BGP community and prefix lists – that are already identified as one of the most common commands – are among the ones experiencing frequent changes.
- In UCB, the commands `distribute-list` and `no passive-interface` account for almost 80% of the most common changes in routing policies over 3 months. The operator explains that UCB recently did an audit on

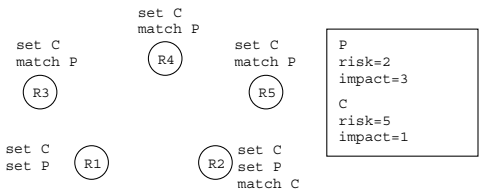


Figure 4: Illustration of the complexity metrics: risk and impact.

their configurations to reduce the number of misconfigurations on OSPF and RIP.

To summarize, router configurations are touched frequently, but each change usually involves very few lines in the file.

4.2 Complexity

Previous sections showed that routing policy definitions rely heavily on route tagging and filtering. This section focuses on the BGP community attribute and introduces two measures to quantify its complexity: its risk of misconfiguration, and the potential impact of an error. We found that a number of communities are recurring in a large fraction of routing policy definitions. In one network, three communities are present in 61% of all eBGP sessions' routing policies. Modifications in any of these communities can thus affect hundreds of sessions.

Intuitively, the risk of misconfiguration of a community increases proportionally to the number of BGP sessions that set this community upon re-advertising a route. The impact of a misconfiguration of a community is proportional to the number of BGP sessions that take actions on routes matching that community. These measures can help to identify the communities that need to be manipulated with care and assess the overall risk and impact the communities in a network.

We now define the two complexity metrics formally. Let B be the set of all BGP sessions in a network, $\Omega(B)$ be size of B, in terms of number of sessions. Let $M_{out}(b)$ be the set of statements m_i that define the export routing policies applied to a session b, and likewise, $M_{in}(b)$ be the set of statements that define the import routing policies on b. Let the predicate $p_{match}(c, M_{out}(b))$ be true if $\exists m_i \in M_{out}(b)$ such that m_i contains a match on the community c, and false otherwise. Similarly, let the predicate $p_{set}(c, M_{in}(b))$ be true if $\exists m_i \in M_{in}(b)$ such that m_i contains a set on the community c. The risk $\mathcal{R}$ and impact $\mathcal{I}$ of a community c are defined as, respectively:

$$\mathcal{R}(c) = \Omega\{b : b \in B \wedge p_{set}(c, M_{in}(b))\} \quad (1)$$

$$\mathcal{I}(c) = \Omega\{b : b \in B \wedge p_{match}(c, M_{out}(b))\} \quad (2)$$

Figure 4 shows an example of the two metrics. The network includes five routers, {R1, R2, R3, R4, R5}, and uses two communities, `P` and `C`. `P` is set

from two routers, {R1, R2}, and matched on three routers, {R3, R4, R5}. Consequently, there are two locations where `P` can be mistakenly set on route advertisements and such mistakes can affect route redistribution from three routers. The risk and impact of `P` are therefore 2 and 3. The other community `C` can be set from all five routers. Thus, the risk=5 is greater than `P`. However, the impact=1 is less since `C` is matched by only one router, R2.

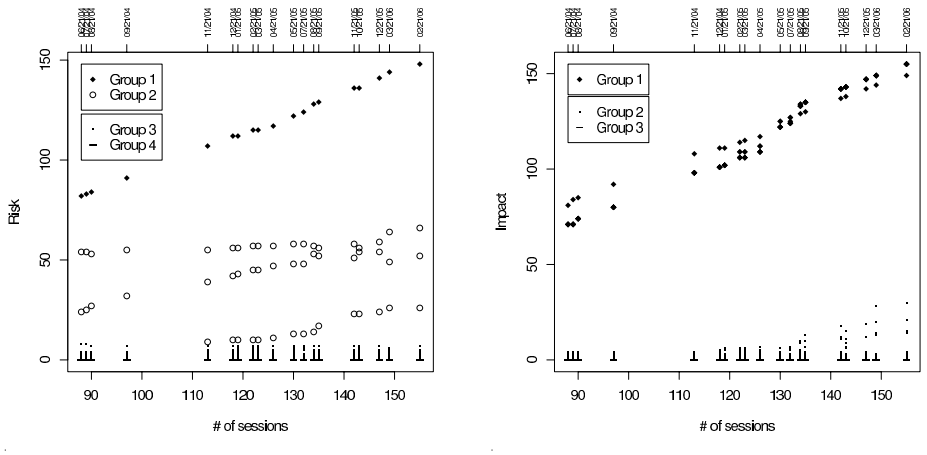
Figures 5 (a, b) depict the risk and impact associated with the communities used in the sessions connecting DC with its peers. They focus on BGP sessions with commodity peers. The impact graph represents a lower bound of the potential consequences of a misconfiguration in communities since communities may be matched in other networks. Similarly, the risk graph represents a lower bound since communities may be set internally or in other external networks. In Figure 5 (a) (resp. b), each dot represents a community. The y-axis represents the number of BGP sessions that sets (resp. matches) that specific community, and denotes the increasing time while the x-axis indicates the total number of BGP sessions between DC and its peers at a specific snapshot, and the date of the snapshot.

Figure 5 (a) shows four groups of communities:

- Group 1 consists of a single community, present in most BGP sessions between DC and its peers. This community is used to mark the routes learnt from peers. Its risk increases over time as the number of peers grows.
- Group 2 comprises of a number of communities indicating the source of the routes. More specifically, they designate the routers where the routes are learnt. Their respective risk also increases with the number of BGP sessions with each router.
- Group 3 contains communities with lower risk. These communities typically uniquely identify the peers (at the level of AS) for routes customization.
- Group 4 indicates a risk level of 0. These communities correspond to the tags that are matched but not set. They comprise the communities that are set to routes originated internally or set in external networks.

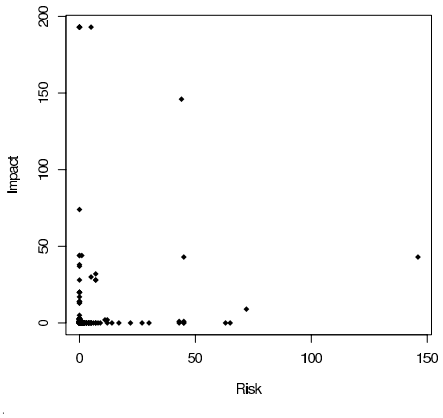
Figure 5 (b) shows the impact of the communities used in sessions between DC and its peers. Three main clusters of communities stand out:

- Group 1 consists of two communities. They are used to specify the routes to filter and the ones to advertise to all the peers. The impact of these two communities increase with the number of peering networks as they need to be present in all corresponding BGP sessions.
- Group 2 is a set of communities used for customization. Certain routes from different customers are advertised to specific classes of peers but not others.



(a) Risk

(b) Impact



(c) Impact vs. Risk

Figure 5: Complexity of communities used in DC with its commodity peers in (a) and (b), and with all its neighbors in (c).

- Finally, Group 3 indicates an impact of 0. We correlated some of these communities between the routing policies in DC and UCB and found that some of these communities are used for inter domain routing. Set in DC, the communities are matched in UCB.

The graph representing the risk and impact of the communities used with customer networks present similar patterns.

Finally, Figure 5 (c) represents the impact and risk of all the communities

used in DC for the latest snapshot (March 2006) of the configuration files. It highlights the existence of two sensitive communities. One presents a risk of 50 and an impact of 150 while the second represent a risk of 150 and an impact of 50. Both communities serve to coordinate routes between a set of customers and a set of peers.

The BGP community attribute was introduced by IETF to facilitate the configuration of complex routing policies. It is highly expressive, and allows an operator much flexibility in its use. However, as discussed in [22, 6], community values are typically defined in an ad-hoc manner and coordinated across routers and autonomous domains manually. The risk and impact of a community may not be directly visible during configuration. We found that the complexity of communities can grow over time, especially when a network expands in size and function. We believe the risk and impact metrics introduced here can help to identify communities within a network that need to be carefully maintained. More importantly, one can evaluate the complexity of a routing policy design in terms of these metrics, and simplify it if necessary before deployment.

4.3 Duplication

The previous section showed that large sets of BGP sessions may perform similar operations when comparing the actions related to communities. This section investigates the degree of duplication between not only communities but subsets of routing policies across BGP sessions. We try to understand how similar routing policies between BGP sessions can be, and if there exists meaningful patterns. The previous section suggested that peers may show certain patterns while customers may display some others. We find that routing policies across routers can present a high degree of similitude: certain networks present more than 400 patterns, present in up to 100% of the routers.

We start by investigating the frequency of appearance of a same community-list and prefix-list definition among routers. We find that up to 91% of the defined community-lists and 54% of the defined prefix-lists may appear in at least two routers. These numbers show that same definitions of both community-lists and prefix-lists are being re-used across routers. Also, 13 community-list and 15 prefix-list definitions were present in more than 25% of the routers of a network. As such, it appears that large fractions of routers re-use common definitions.

To further assess the level of duplication across routing policies defined in the routers of a network, we introduce a duplication measure which includes some limited form of semantic equivalence. We want to compare the degree of similitude among subsets of routing policies. As described in Figure 1, routing policies are typically defined as a sequence of conditions and actions. Routes matching a set of conditions, receive the specified actions. Therefore, we convert each command line of a routing policy definition into a token, taking into consideration the direction of the routing policy (in, out) and the action performed on the route (e.g. permit, deny). We de-reference the names of the lists to detect common subsets of routing policies across routers independently of the

Network name	Total num. of sessions	Num. of patterns	Pattern sizes (min,med,max)	Cluster sizes (%) (min,med,max)
DC	244	407	(1,18,34)	(5,11,81)%
HPR	68	152	(1,23,40)	(6,25,100)%
UCB	11	8	(1,5,17)	(2,2,72)%

Table 3: Duplication of routing policies across the three networks.

assigned names. As an example, the routing policies of Figure 1 are converted into the following five tokens:

```
in_permit_set_localpref_100
in_permit_set_comm_100_1
in_permit_set_comm_100_2
in_permit_set_comm_100_3
out_deny_match_comm_100_4
```

However, we do not expand regular expressions. We believe the results are still indicative of the degree of duplication of routing policies among routers. Because we do not detect common patterns among regular expressions that are syntactically different but semantically equivalent, the results present a lower bound of the existing duplications.

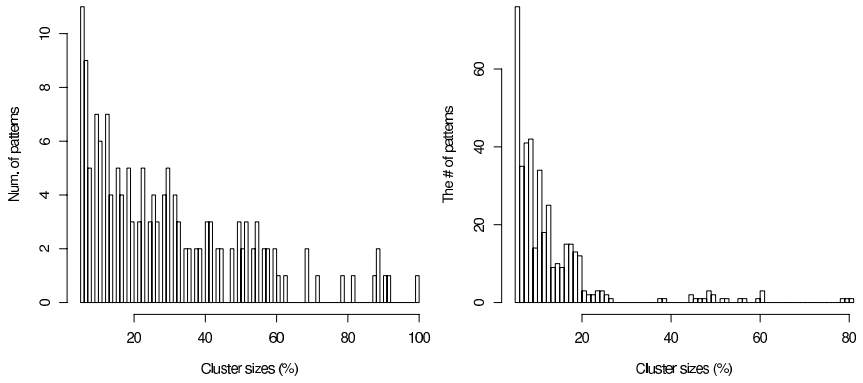
While each router vendor supports a wide range of conditions ("match statements") and actions ("set statements"), we focus on four primitives: prefix list, AS path, community and local preference. This decision is motivated by the fact that these four primitives are the most commonly used in the match and set statements of the analyzed configuration files. Such analysis allows us to detect how prevalent a same command is, in the existing routing policies.

A token represents an action in a routing policy definition. We call a set of tokens (i.e., actions) shared by at least k sessions, a pattern. We define $k = \max(0.05 \times \Omega(B), 2)$, where $\Omega(B)$ is the total number of sessions. The number of patterns is indicative of the degree of similitude between routing policies. The size of the clusters represents the number of sessions presenting a specific pattern. Finally, the size of each pattern illustrates the number of common actions in the shared policies. Table 3 summarizes the results for the three networks. The study focuses on the routing policies applied to the eBGP sessions. We did not consider the internal routing policies.

The analysis reveals a large number of patterns in each network. As an example, DC presents more than 400 patterns. Figure 6 further presents the distribution of the discovered patterns. We found that:

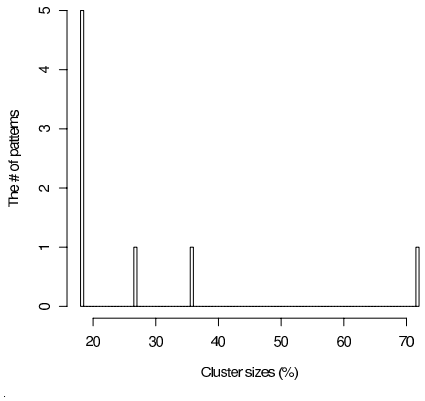
- Most patterns are shared by small clusters of eBGP sessions. However, because of the large number of eBGP sessions in some networks, a cluster size of 10% still includes more than 20 eBGP sessions.

- A number of patterns are present in more than half of the eBGP sessions. In HPR, some routing policies are applied to all eBGP sessions. These



(a) HPR

(b) DC



(c) UCB

Figure 6: Duplication of routing policies across the three networks.

actions are related to the filtering and distribution of routes through communities.

- Patterns can be large, including up to 40 common actions. The analysis of these patterns reveals that most of the actions present in large patterns are related to communities. Such results confirm the complexity of the configuration of communities presented in the previous section. We observed that in many instances, groups of actions related to communities are performed together.

The number of discovered patterns and the number of BGP sessions presenting a same pattern suggest that common subsets of routing policies are

typically applied across large subsets of BGP sessions. This required degree of consistency in turn implies that the risk of misconfiguration is elevated and exacerbated by current network management techniques where each router is configured manually and separately.

5 Misconfigurations in Routing Policies

In the previous section, we find that the configurations of routing policies exhibit definite and complex patterns, both within a single router and across multiple, if not sometimes all, routers in a network. As echoed by others, configuring routers is a tedious and error-prone task. Identifying mistakes in router configurations is tricky however. What constitutes an error sometimes depends on the network – what is an error for one network can be acceptable for another. Local rules of a network can be complicated and usually not captured by universal rules set forth by common best practices.

In this section, we apply data mining on router configurations to infer local, network-specific routing policies and detect potential errors that deviate from the inferred policies. 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 also be used to identify errors in network configurations, as we have observed network elements often share common configurations.

In the three networks we studied, route tagging is widespread. In particular, the use of BGP communities is complex and can lead to high risk and impact on a network. A single error in the configuration of a community can result in inaccurate route advertisements which can affect both the internal network and external neighbors. As we described previously, there is a substantial amount of duplication in the configurations of routing policies, especially in communities. Data mining can be used to find such patterns.

We also find that the commands `distribute-list` and `no passive-interface` are among the most common and frequently modified commands in the configurations of UCB. The operator explains [2] that UCB's policy on OSPF is to have all interfaces passive by default and explicitly use the command `no passive-interface` for backbone interfaces. Thus, any non-backbone interface that is not passive is misconfigured. Similarly for RIP, the network has a number of subnets with user devices using RIP for their default paths. On non-backbone interfaces, the local policy mandates a `distribute-list out` command so that only the default route would be sent out, and a `distribute-list in` command to ensure routers only accept valid announcements originated by the user devices. If any of the above commands is missing in this context, RIP is considered misconfigured. Again, data mining can discover patterns like these.

We describe our data mining method and the results on applying it on the three networks next.

5.1 Data Mining with Minerals

In a previous paper [15], we proposed Minerals, an approach to detect errors in router configuration files using association rules mining [14]. Here, we summarize Minerals, and briefly discuss its strengths and weaknesses next.

5.1.1 Association Rules Mining

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 $S(X \wedge Y)/S(X)$.

In Minerals, rules with high confidence are considered reflections of local network policies. Configurations that deviate from these rules are identified as potential misconfigurations because they do not comply with the inferred policies. We call these *violations*. The underlying assumptions are that there exists common configurations across routers, and the number of properly configured functions is larger when compared to the number of their misconfigured counterparts – otherwise, the network would not operate properly.

Minerals can be applied to a network without customization and operator involvement. Note that Minerals does not attempt to understand the meanings behind configurations, for example, it does not know the role and function of a BGP community tag in a network, nor whether a prefix list is for filtering bogon prefixes. While this lack of understanding from Minerals can be viewed as a limitation, this trait is beneficial as it allows Minerals to be applicable to a broad class of configuration commands.

The strength of Minerals is also its weakness. Since Minerals searches for common patterns in a network's configurations, errors in unique configurations may not be detected. Minerals may also point out valid, non-conforming configurations, such as a customer with a special arrangement, as violations, leading to false alarms. Also, Minerals cannot find design and logic errors which violate general network goals, e.g. incorrect packet filters preventing subnets from communicating.

Minerals is a general approach and can be applied to routing policies specified for different routing protocols such as BGP, OSPF, IS-IS, OSPF, etc. In this paper, we focus on BGP routing policies.

5.1.2 Pre-Processing

In data mining, pre-processing the input data is essential to remove irrelevant information that can confuse a learning algorithm and yield useless results [14]. Experiments have shown that data mining is sensitive to the input data, and there is no exception here. Below, we describe how to pre-process the configurations of routing policies into a format suitable for data mining. Note that in our implementation, the pre-processing step is automated and is general so that almost all of it can be executed on different networks without modification. We discuss pre-processing in great detail in our previous paper [15].

There is a plethora of commands to configure routing policies. We choose a small subset of them based on our observation that a significant portion of routing policies are related to route tagging and filtering. In particular, we focus on BGP communities, AS Path, prefix lists and local preference. We distill these commands to remove vendor-specific syntax.

We first parse router configurations from vendor specific syntax to a common intermediate representation. For each reference to a list (i.e. prefix-list, community-list and aspath-list), we expand the reference to the list definition. This allows us to focus on the values of the lists and not the names assigned to the lists. For example, `sanity` is used to name a prefix list containing unallocated IP address space in one router but `bogon` is used in another router.

We then distill each action of a routing policy into an attribute. In data mining terminology, an attribute is a feature of data, and a set of attributes characterizes the data. Similar to the way tokens are generated to compute the duplication factor in §4.3, for each line of a routing policy applied to a BGP session, we extract the direction of the policy (incoming or outgoing), the conditions being matched (community, AS path, prefix list) and the action taken (set community or local preference, permit or deny). For example, the `route-map from_dora` defined in Figure 1 would be converted into the following five attributes:

```
in_permit_set_localpref_100
in_permit_set_comm_100_1
in_permit_set_comm_100_2
in_permit_set_comm_100_3
out_deny_match_comm_100_4
```

An attribute can take on different values. In our case, an attribute takes on a boolean value of either 1 or 0: 1 if a routing policy contains the action represented by the attribute, and 0 otherwise. Thus, the attributes above would take on the value 1 for the BGP session the routing policy is applied on.

In the last pre-processing steps, we extract whether MD5 is applied to the session and use the AS number of the peer in a BGP session to determine whether the session is an external or internal one.

In our current implementation, the pre-processing step only yields a partial semantic comparison of lists. Regular expressions are not expanded. Two regular expressions may differ syntactically but have the same semantics, e.g. (100|200) and (1-2)00. Vendor-specific command syntax also presents additional

challenges. For example, to specify an AS path that includes the AS number 100, IOS would define it as _100_, where as JUNOS would define it as ".* 100 .*". Our representation also does not consider the ordering of lines within a routing policy. Just like a program can be written in many ways, one can achieve the same routing policy with different configurations.

Despite these limitations in our current implementation, we believe it is acceptable for the following reasons. First, to keep problems at a minimum, networks are commonly configured by a small group of highly skilled experts. Second, existing configurations are often used as templates to configure another router or another routing policy. In the previous section, we show that there are substantial amount of "cut-and-paste" in the three networks we studied.

Later, we show that Minerals found a number of errors which are confirmed by the network operators.

5.1.3 Post-processing

After pre-processing the configurations and running our association mining algorithm on them, we apply a post-processing step to filter the results. The goal is to increase accuracy of detection through reducing false alarms.

The first post-processing step attempts to identify whether the policies for a group of neighbor routers in the same AS are non-conforming but intentional. Even though policies for a specific neighbor AS are not consistent with policies with other ASes peering with a network, they may not be misconfigurations because it is not uncommon for a network to customize policies for each neighbor AS. We assume that if policies are consistent across all BGP sessions to the same AS, the violation highlighted by Minerals is likely not a misconfiguration. This post-processing works as follows:

1. For each reported violation, denoted by BGP session s_0 and associate rule $((a = a_i \wedge b = b_j \wedge \ldots) \rightarrow m = m_k)$, we find all BGP sessions $s_1, s_2, \ldots, s_n$ that are peering with the same (AS number, peer-group), and the values of the attributes $a, b, \ldots, m$ for each session.

2. If each attribute has the same value across all BGP sessions $s_0, s_1, s_2, \ldots, s_n$, the violation is demoted.

Note that the attributes $a, b, \ldots, m$ above do not necessarily represent the complete routing policy applied to a BGP session. In Minerals, a rule can be a subset of a routing policy, and the rule passes the support and confidence thresholds. In other words, $a, b, \ldots, m$ are attributes that are reported in the detected rule. We do not assume all routing policies to the same AS must be identical. In fact, in some configurations, each router can add a distinct community tag thus rendering every route-map unique if we compare route-map in its entirety. Minerals does not highlight each of these route-maps as an anomaly, because it only focuses on commands that appear frequently in routing policies.

There are several ways to demote a violation. In our current design, we simply discard the violation. However, this may prevent some errors to go undetected. A misconfiguration can be present on all BGP sessions to the same AS because of a cut-and-paste operation. There is an inherent tradeoff between the detection and false alarm rates of Minerals. Several operators have told us that they want to know about all inconsistencies in their configurations, even though some are highly likely to be valid. Another way to demote a violation is to lower its priority when Minerals presents its results to the network operator.

The second post-processing step is related to MD5 security. Best common practice says it should be enabled on eBGP sessions. However, in practice, some older routers do not support this feature well and operators sometimes disable it for performance or stability reasons. If Minerals reports a violation of missing MD5 protection on a BGP session, this post-processing step checks to see if any of the two following conditions is satisfied: i) the offending router is having sessions with other ASes that require MD5, but all sessions from this router have MD5 disabled, or ii) all sessions to the other end-point of the session have MD5 disabled. If so, we assume the local policy says to disable MD5 on this router or the other end-point, and demote the violation.

The last post-processing step identifies simple BGP sessions, which we define as sessions which either use default routing, or block all incoming or outgoing route advertisements. If Minerals raises a violation on a simple BGP session, we demote it.

5.2 Results

We data mine configurations files of DC, HPR and UCB using Minerals. We set `min_conf` to 95%. Table 4 summarizes the results in three categories: *Confirmed Errors*, *transitions* and *false alarms*. *Confirmed Errors* are misconfigurations that are confirmed by the network operators. *Transitions* represent violations that reflect policies that are undergoing change. While such violations are not always errors, they should be brought to the attention of network operators. If not updated in a timely manner, these cases can become errors. For example, in one case, an outdated community that was not removed led to a customer filtering out an intended route. *False Alarms* are violations raised by Minerals that are non-conforming but correct configurations.

- 50% of the errors are related to missing communities. These misconfigurations affected the redistribution of routes, the aggregation of prefixes and the accounting of certain traffic.
- Few typos were found in the definitions of communities. These errors prevented the intended actions to be performed on different routes.
- A number of sessions lacked either import or export routing policy.
- The detected errors also included missing prefix-lists, both in the incoming and outgoing directions.

	Confirmed errors	Transitions	False Alarms
Num. of reported BGP sessions	30	29	16

Table 4: Summary of Minerals results.

- Few sessions lacked MD5 security violating the local policies

We describe the three categories of results next in more details. All results are anonymized.

5.2.1 Confirmed Misconfigurations

Missing communities. In §4, we show that route tagging constitutes one of the most common commands and changes in router configurations. We also discuss the complexity in the configurations BGP communities. 50% of all the confirmed errors are related to missing communities.

A number of these errors are accidental omission of community tagging, which lead to several undesired outcomes. In one case, internal routes are leaked to external networks. In another case, prefixes to be aggregated are to be tagged with specific communities and filtered out at the egress routers. The absence of the communities on certain routes affected route aggregation and resulted in the advertisement of some of the more specific prefixes. In one instance, missing communities prevents routes from being advertised to a set of peers.

One interesting error impacted accounting. As communities control route announcement, a missing community prevented a more specific route from reaching an edge router that performs accounting. In this case, the lack of proper tagging does not affect connectivity, but breaks billing for the network.

Missing export policy to internal peer group. *Rule:* Routers A and B are route reflectors in the same cluster, and C is their route reflector client. C is a low-end router with limited memory and processing capabilities. The local network policy is that A and B should advertise only a default to C, and C should also filter incoming route announcements.

While the session between A and C contains the export policy, the session between B and C does not. The lack of export policy does not impact the connectivity since C is also filtering incoming routes. However, the error means B is sending unnecessary announcements to C. The operator confirmed this as an undesired configuration because of the capability of C. It would be preferable to install export policy at B as in A.

Missing outgoing filter. *Rule:* the local network policy treats a set of neighbor ASes differently and does not advertise certain routes to them. The outgoing route-maps to these networks therefore typically starts with the following:

```
route-map to_abc deny 10
 match community nwB_no_export
```

The route-map to one of this special networks mistakenly lacks this match statement. A strength of Minerals is that it can detect such outliers without knowing the meaning of the statements involved. Minerals does not understand the function of the community list `nwB_no_export`.

Missing incoming prefix-list. *Rule:* A local network policy that requires a prefix-list to be applied on all incoming external BGP sessions. The prefix-list turns out to be a sanity check that discards route announcements to private IP address. Two BGP sessions in this network violates this policy.

One version of a correct route-map:

```
route-map from_abc permit 50
 match ip address prefix-list sanity
 match community ...
 set local-preference 90
 set comm-list ... delete
 set community ... additive
```

One version of the violations:

```
route-map from_bcd permit 50
 set local-preference 100
 set comm-list ... delete
 set community ... additive
```

Note that although the two route-maps above are different in several places – different local preference values, match community statement in the first but not the second – Minerals is able to detect similarities in parts of the route-maps and points out the missing prefix-list statement. As mentioned before, Minerals do not require routing policies to be identical in order to find inconsistencies because association rules mining can work on subsets of attributes defining a policy.

Missing outgoing prefix-list. *Rule:* Similar to above, but a local network policy that requires a prefix-list to be applied on all outgoing external BGP sessions. The prefix-list is also a sanity check. In IOS, a `match ip address prefix-list sanity` is used. Two BGP sessions are missing this statement.

Incorrect community-list definitions. In a number of routers, a community-list is incorrectly defined:

```
ip community-list 50 permit permit regexp
...
```

The repetition of the word "permit" is an error. While a syntax-based configuration checker can also detect this type of errors with a specific rule looking out for "permit permit", Minerals allow us to detect syntactical abnormalities using the same algorithm. In this case, a large number of BGP sessions have this error and a few sessions have the intended definitions. Minerals highlighted the error as the rule and the violations are actually the correct configurations.

```
ip community-list 60 permit regexp
...
```

Missing MD5 security. Minerals found a large number of violations on missing MD5 security. The second post-processing step described in §5.1.3 removed a substantial portion of these violations. In one case, the violations are sessions to ASes that preferred not to use MD5. All the sessions to those ASes are found with MD5 disabled and therefore removed. Another case involves a specific router. The post-processing step found that MD5 is disabled on every session involving that router. It is likely that router does not support MD5. It turns out some ASes resist turning on MD5 because it add delays to session re-establishment after a reset.

The rest of the violations are sessions with neighbor networks that require MD5. These sessions were not removed by the post processing step, and are confirmed as errors by the operators.

Typos in community list definitions. In one router, there is a typo in a community list definition. The local AS number is 123, and the typo is in one of the digits in number, making it 124:

```
community core-pref members [123:100 124:101];
```

Missing export policy to external AS. In one network, two eBGP sessions are missing export policies.

5.2.2 Transitions

Outdated communities. One of the networks we studied went through a routing policy change. The tagging of routes coming from commodity peers has been modified to allow more flexible customizations. We found that sessions that are not updated could lead to undesired effects. In one session, communities are still set according to the previous policy and the routes announced from this peer are filtered by a customer. As there is another up-to-date session to the same peer, the customer is still able to reach the peer's network. However, if the up-to-date session is disabled for maintenance or other reasons, the customer would experience a loss of connectivity to that peer's network.

Distribution of internally originated routes. One of the analyzed networks advertised its internal routes to a selected set of neighbors through a `match prefix-list internal-routes` command in which the prefix-list listed all the internal prefixes. This configuration is being changed and internal routes are now tagged with a specific community at origination. The configurations are updated to include a `match community` command to allow the propagation of internal routes to the desired parties. We found a number of BGP sessions still have the initial configurations. When `match community` is accidentally omitted in some of the sessions, some internal routes may be missing from the route

advertisements. More specifically, internal routes not listed in the outdated prefix-list can not be forwarded to the neighbors.

Transit service. In one network, Minerals found a research peer is receiving transit service for a wide range of prefixes. Usually, only customer routes are advertised to peers. This configuration does not violate the local network policies as it is a special case arrangement for a certain type of peers, but it is not a permanent situation.

Inconsistent community-list definition. A community list was defined in the same way in 18 routers, but was slightly different in another router. It turn out that the definition is being updated but the modification is not complete when we carried out our evaluation.

Common definition (present in 18 routers)

```
ip community-list 190 permit 100:6550[1-9]
ip community-list 190 permit 100:6551[0-4]
```

Outlier (present in one router)

```
ip community-list 190 permit 100:6550[1-9]
ip community-list 190 permit 100:6551[0-9]
```

The outlier in this case is the intended definition whereas all other definitions should be updated.

5.2.3 False Alarms

The false alarms in our results mainly consist of highly customized routing policies. In one network, there are a number of neighbor ASes that are multi-homed only wanting to accept certain routes. Knowing that the configurations to these neighbors are unique and significantly different from the rest of the configurations, we could have removed them from our evaluation. However, we included them. A number of errors are detected on the sessions to these ASes, such as accidental omission of prefix-list.

6 Related Work

Much research has been conducted in the area of routing policies. We differentiate research related to this paper into three main thrusts: routing policies and their complexity, frequency of errors in network configuration, and misconfiguration detection mechanisms.

6.1 Routing Policy Design and Complexity

Maltz et al. [17] developed several methods to reverse engineer a network's routing design from its router configuration files. Their techniques allow to

abstract and represent the flow of routing information within a network and the interaction between routing processes. Instead, our work focuses on how operators enforce the desired policies.

Quoitin et al. [19] correlate information from the RIPE whois database and BGP UPDATE messages from RIPE RIS and RouteViews, and found that BGP communities are widely used to implement a variety of policies. Their focus is on BGP communities and their results led to the design of a more structured form of redistribution communities. Our work differs in two ways. First, we consider the configuration of routing policies in general, and other features of configuration commands. Second, the use configuration files allow us to be at the source of routing policies, which details are not exposed in routing messages.

Broido et al. [4] introduced a framework to analyze the complexity and redundancy of global routing policies. However their motivations and our objectives differ. Broido focused on the complexity mainly in terms of the number of prefixes in BGP tables and aimed at reducing the size of BGP tables to reduce the CPU and memory required at the routers. Instead, our goals consist in understanding current routing policies as a whole for better network management.

6.2 Network Misconfiguration Studies

Wool [21] studied the firewall rule sets in a number of networks and discovered that most of the firewalls included errors. Even though his analysis also shows the importance and the complexity of configuration and management, Wool's work concentrates on firewalls whereas our study focuses on routing policies.

Mahajan et al. [16] measured the frequency of BGP configuration errors by looking at BGP messages from the Oregon RouteViews servers. The authors found that misconfigurations were frequent, and over the course of 21 days during which the experiment was conducted, the authors estimated that "close to 3 in 4 of all new prefix advertisements were results of misconfiguration." While their approach presents the advantage of not requiring access to network configurations, it only allows to analyze errors that are visible in BGP advertisements.

6.3 Misconfiguration Detection

The work of El-Arini and Killourhy [8] is perhaps the closest to Minerals. They use a set of Bayesian based algorithms to detect statistical anomalies in router configurations. While the goal is similar, the approaches differ. Their algorithms focus on the frequencies of the command lines, and assume that each command line is independent and do not draw any relations between different commands.

A number of solutions have also been proposed to deal with the router misconfiguration problem. RAT [20] compares a router's configuration file with the recommendations set forth by the NSA [18]. Based on the differences, it assigns a score to indicate the security level of the router in question. Feldmann and Rexford [10] are the first to propose to detect misconfiguration through the

parsing and analysis of router configuration files. rcc [9] detects 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 [10, 9] are rule-based approaches in which rules or policies need to be known *a priori* and to be provided in advance. As discussed previously, rule-based approaches check for violations to universal rules, unless network-specific rules are defined on a case-by-case basis.

EDGE [7] 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.

7 Concluding Remarks

In this paper, we analyze the configurations of three networks over a period of 3 to 21 months. First, we find that routing policies represent a dominant part of configurations. Second, routing policies rely heavily on route tagging and filtering. Finally, in the observed networks, the commands related to these features are frequently modified, almost daily. Interestingly, 80% of modifications involve 15 lines or less. This suggests that most modifications are incremental.

The analysis of route tagging reveals that communities are complex to configure. They can present elevated levels of risk and impact. Some communities can affect up to hundreds of eBGP sessions, making them difficult to configure and highly prone to errors. Both risk and impact grow with time as the number of peers increase.

We also discover a high degree of similarity across routing policies within a network. We find more than 400 distinct patterns in one network. Many patterns are applied to more than 50% of the eBGP sessions, and up to 100% of them in some networks. These numbers suggest that there exists a high degree of consistency among routing policies within a network. The analysis of these discovered patterns shows that they are indeed meaningful, and inconsistency could often indicate a misconfiguration.

Based on these observations, a misconfiguration detection scheme is developed and applied to router configuration files. The algorithm is based on data mining. It searches for patterns and raises inconsistencies. Such approach is general and does not require per-network customization. We evaluate our scheme on three different networks including both transit AS and stub AS. Our method successfully infers local network policies and detects misconfigurations The generality of our solution allows us to detect a number of error types ranging from erroneous communities, to missing prefix-list.

Interestingly, more than half of the errors are related to communities. Another significant observation lies in the proportion of violations related to transitions i.e., routing policies undergoing change. These instances can present a threat and negatively impact the operations of a network if not updated in a timely manner.

We are currently exploring the application of time-series analysis to router

configuration files. The findings in this paper reveal that a number of communities need to be added every time a session to a customer or a peer network is created. We believe time-series analysis can help in two aspects: first, to discover similar patterns over time and help operators in the configuration of routers, and second, to enlarge the scope of errors we can detect.

In conclusion, we believe next-generation policy language and routing control platform should allow network operators to do frequent and flexible tweaking, besides specifying their intent and constraints. Although one might argue tweaking is an artifact of a broken architecture, not a necessity, operators are cautious and would like to roll out changes in their network in stages, in order to limit impact.

Acknowledgments

This work was funded in part by KISA, MIC, Samsung, ARO and Carnegie Mellon CyLab. We would like to express our gratitude to Ken Lindahl and colleagues at CNS for giving us access to Berkeley's configuration files, for comments and suggestions, and for educating us about network operations.

References

[1] Corporation for Education Network Initiatives in California (CENIC). http://www.cenic.org/.

[2] Private communication with Ken Lindahl, April 2006.

[3] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. *Routing Policy Specification Language (RPSL)*, 1999. RFC-2622.

[4] A. Broido and k claffy. Analysis of Route Views BGP data: policy atoms. In *Network-Related Data Management (NRDM) Workshop*, 2001.

[5] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a Routing Control Platform. In *NSDI*, May 2005.

[6] M. Caesar and J. Rexford. BGP policies in ISP networks. In *IEEE Network Magazine, Special Issue on Interdomain Routing*, 2005.

[7] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford. The Cutting EDGE of IP Router Configuration. In *HotNets-II*, Boston, MA, November 2003.

[8] K. El-Arini and K. Killourhy. Bayesian Detection of Router Configuration Anomalies. In *Sigcomm Workshop on Mining Network Data*, 2005.

[9] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *NSDI*, May 2005.

[10] A. Feldmann and J. Rexford. IP Network Configuration for Intradomain Traffic Engineering. *IEEE Network Magazine*, 2001.

[11] L. Gao. On inferring autonomous system relationships in the Internet. In *IEEE Global Internet Symposium*, November 2000.

[12] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A Clean Slate 4D Approach to Network Control and Management. *ACM Sigcomm CCR*, Oct 2005.

[13] T. Griffin, A. Jaggard, and V. Ramachandran. Design Principles of Policy Languages for Path Vector Protocols. In *Sigcomm*, August 2003.

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

[15] F. Le, S. Lee, T. Wong, H. S. Kim, and D. Newcomb. Minerals: Using Data Mining to Detect Router Misconfigurations. Technical Report CMU-CyLab-06-008, May 2006. Under submission. Available at: https://www.cylab.cmu.edu/default.aspx?id=2200.

[16] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP Misconfiguration. In *Sigcomm*, August 2002.

[17] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg. Routing Design in Operational Networks: A Look from the Inside. In *Sigcomm*, September 2004.

[18] 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.

[19] B. Quoitin and O. Bonaventure. A survey of the utilization of the BGP community attribute, February 2002. Internet Draft (expired).

[20] The Router Audit Tool (RAT). http://www.cisecurity.org/bench_cisco.html.

[21] A. Wool. A Quantitative Study of Firewall Configuration Errors. *IEEE Computer*, June 2004.

[22] M. Yannuzzi, X. Masip-Bruin, and O. Bonaventure. Open Issues in Interdomain Routing: A Survey. *IEEE Network Magazine*, Nov. 2005.