A Firewall for Routers: Protecting Against Routing Misbehavior

Ying Zhang Z. Morley Mao Jia Wang
University of Michigan *AT&T Labs–Research*

Abstract

In this work, we present the novel idea of route normalization *by correcting on the fly routing traffic on behalf of a local router to protect the local network from malicious and misconfigured routing updates. Analogous to traffic normalization for network intrusion detection systems, the proposed RouteNormalizer patches ambiguities and eliminates semantically incorrect routing updates to protect against routing protocol attacks. Furthermore, it serves the purpose of a router firewall by identifying resource-based attacks against routers. Upon detecting anomalous routing changes, it suggests local routing policy modifications to improve route selection decisions. Deploying a RouteNormalizer requires no modification to routers if desired using a transparent TCP proxy setup.*

In this paper, we present the detailed design of the RouteNormalizer and evaluate it using a prototype implementation based on empirical BGP routing updates. We validate its effectiveness by showing that many well-known routing problems from operator mailing lists are correctly identified.

1 Introduction

> I would stress that all of these things, particularly prefix hijacking and backbone router "ownage", are real threats, happening today, happening with alarming frequency. Folks need to realize that the underground is abusing this stuff today, and has been for quite some time.
>
> –Rob Thomas quoted by David Meyer at NANOG 28, June 2003

This is a quote given by David Meyer, a well-known network researcher and network operator, at the North American Network Operators' Group (NANOG) Meeting in 2003 [33]. It highlights the urgency to better protect the Internet routing system, providing the main motivation for our work.

The Internet originated from a research network where network entities are assumed to be *well-behaved*. The original Internet design addresses physical failures well, but fails to address problems resulting from misbehavior and misconfigurations. Routers can misbehave due to misconfigurations [29], impacting network reachability. Today, the Internet has no robust defense mechanisms against misbehaving routers, leaving the routing infrastructure largely unprotected [34]. One of the most widely known and serious misconfiguration occurred in 1997, when a customer router at a small edge network by mistake advertised a short path to many destinations, resulting in a massive blackhole disconnecting a significant portion of the Internet [10]. This example illustrates the need for an easily deployable protection mechanism to prevent local forwarding decisions from being polluted.

For our purpose, we define the *control plane* to be the Internet routing layer, and the *data plane* to be the packet forwarding layer. There is an inherent trust relationship in today's routing system: a router assumes routing updates from its neighbors are correct. However, router misconfigurations [29], attacks [30], and inherent routing problems [21] often render this assumption incorrect.

Given the lack of security in today's routing protocols, both the research and the network operator community have already proposed solutions such as SBGP [40] and SoBGP [35], which require routing protocol modifications. However, we have witnessed a rather slow deployment. Furthermore, most of them do not eliminate the possibility of router misconfigurations and their associated impact.

We take a different approach by posing the question of *what individual networks can do locally to protect against routing misbehavior from external networks*. Even if future routing protocols have enhanced security mechanisms, there is still a need to be defensive against routing attacks from noncooperative networks or misconfigurations. Furthermore, there exist inherent ambiguities in routing protocols that require rectification to proactively prevent unexpected behavior due to implementation variations. We propose that networks proactively *correct* routing updates locally through a *RouteNormalizer (RN)*, which logically sits on the data path between the local router to be protected and the remote router whose updates may be untrustworthy. Our work fits perfectly with the recent proposal of logically centralized routing architecture such as RCP [12] to improve routing control. Such a platform acts as a firewall for the local router by identifying and preventing routing attacks using anomaly detection. Taking advantage of local information such as local routing policies, local address information, relationship with neighboring Autonomous Systems or ASes help more accurately detect routing attacks directly impacting the local AS.

Unlike protocols such as SBGP, our approach requires no changes to routing protocols or router configurations. We summarize RN's main functionality: (i) Identify and correct anomalous routing updates. (ii) Identify and mitigate routing attacks. (iii) Mitigate routing instability by dampening routing updates. (iv) Perform load management by rate-limiting updates. (v) Emulate features not available on local routers, *e.g.,* graceful restart [39]. One of our novel contributions lies in applying routing anomaly detection to influence routing decisions so that routes selected are more likely correct.

The RouteNormalizer is a general platform for correcting routing updates for any routing protocols. In this work, we focus on the interdomain routing protocol – BGP (Border Gateway Protocol [37, 24, 22]) given its importance to the well-being of the Internet and that its routing information mostly arrives from external untrusted networks. BGP is a *path vector* protocol, as the AS_PATH attribute contains the sequence of ASes of the route. Each BGP update contains path attributes such as NEXT_HOP and ORIGIN, some of which are mandatory. BGP is *incremental*, *i.e.,* every BGP update message indicates a routing change. In addition, BGP is *policy-oriented*: routers can apply complex policies to influence *best* route selection for each prefix and to subsequent route propagation.

We summarize our main contributions: (1) Developed a platform to perform BGP traffic normalization, enabling incrementally deploying new router functionality. (2) Improved on existing router functionality, *e.g.,* max-prefix-limit. (3) Proposed the use of routing anomaly detection to achieve robust routing. (4) Improved routing anomaly detection by exploiting local network information. (5) Performed the first extensive correlation between NANOG emails for routing related complaints with BGP data.

The rest of the paper is organized as follows. We first present the architecture design of the RouteNormalizer in Section 2. Section 3 describes the deployment scenarios. We describe our prototype in Section 4. We show the effectiveness of the RouteNormalizer using empirical BGP data in Section 5. Finally, we cover related work and conclude.

2 RouteNormalizer Architecture

In this section, we describe the architectural design. We refer to the *local router* as the router under the protection of the RouteNormalizer in the same AS. The RouteNormalizer can correct traffic on behalf of several local routers. The *remote router* refers to the other router in the BGP session, typically not within the same AS as the local router.

As illustrated in Figure 1, the RouteNormalizer takes several optional input data such as the local router's policy configuration and real-time BGP feeds from external sources. The RouteNormalizer can be configured to observe all the traffic destined to the local router's BGP port 179, as well as traffic originated by the local router to the remote router. As output, in addition to the "normalized" BGP traffic, it generates alarms and suggestions for policy modifications for the

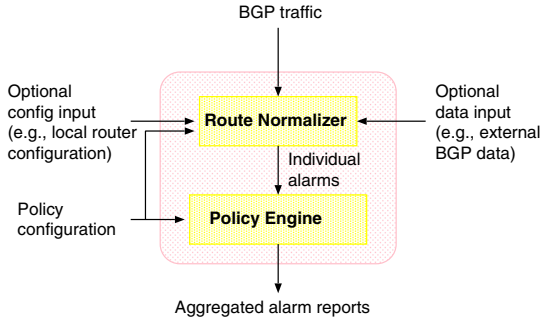

Figure 1. The RouteNormalizer framework.

local router. Note that the RouteNormalizer performs normalization on BGP traffic both destined to as well as originated from the local router. We focus on the former case here.

2.1 Functionality overview

We first highlight its design principles. (1) Basic checking to ensure protocol semantic correctness. (2) Make use of *local network information* such as local network addresses. (3) Take advantage of external information to assist route anomaly detection. (4) Assume dominant history behavior is mostly correct. (5) Use anomaly detection to influence route selection: be conservative by avoiding anomalous routes.

A summary of all functionalities is shown in Table 1. Arguably, some of these can be performed directly by routers using route filters for example. However, route filters are usually signature based, only protecting against known attack patterns. Nor do they have any anomaly detection or data correlation capability. Routers usually have limited memory to perform such filtering, and such incurred overhead may impact regular forwarding behavior. Furthermore, as we explain below, the RouteNormalizer also acts as a firewall to prevent potential router OS bugs from being exploited. For known router OS problems such as the recently announced DoS vulnerabilities in Cisco IOS [43], the RouteNormalizer can act as a filter before patches are applied. Next, we illustrate each category using examples.

2.2 Fix violations of BGP semantics

The RouteNormalizer performs simple checks for detecting violations of BGP semantics in routing updates. Routers may react differently to such updates depending on their implementations. In the ideal case, they would drop such updates and send back an error message. In some instances, these routes may actually be selected as the best route for forwarding; however, packets may not reach the destinations due to the violation of BGP semantics.

Routers from different vendors running distinct software versions may exhibit dissimilar default behavior, possibly leading to inconsistent routing decisions in a single network. As a result, simply enforcing uniform routing configurations across all routers in the network may not be sufficient. Moreover, unexpected BGP updates may also lead to router OS crashes (*e.g.,* [7]). Thus, a platform such as the RouteNormalizer that dynamically detects routing problems is very useful.

Category	Description	RouteNormalizer actions (also suggest improved routing policies)	Existing router implementation
Fix violations of BGP semantics	Incorrect attribute values: *e.g.,* AS loops	Modify, drop updates, generate warnings	Configure with route filters.
	Attributes with private information		
	Missing mandatory attribute values		
Fix violations of routing policies	Export policy violations	Drop updates, generate warnings	Configure with route filters, may require external information.
	Nexthop violations	Modify, drop updates, generate warnings	
Detect routing anomalies	Anomalous routing behavior	Drop updates, generate warnings	Not available, difficult to implement, requires external information.
	Routing inconsistency	Drop updates, generate warnings	
	Local and remote address hijacks	Modify, drop updates, generate warnings	
Load management and instability mitigation	Mitigate load due to identical routing updates	Drop duplicate updates	Partially implementable, RouteNormalizer enhanced existing functionality.
	Mitigate against router DoS attacks	Filter BGP attack traffic, delay updates	
	Mitigate instability of flapping prefixes	Emulate route flap damping, delay updates	
	Mitigate instability of session resets	Emulate graceful restart, delay updates	

Table 1. Functionality of the RouteNormalizer.

1. AS routing loops. This is an example of ambiguities in BGP routing protocol specification. It is recommended that routes containing loops should not be used; however, in practice we still observe such routes due to lack of enforcement. In some rare instances, routing loops are allowed in the AS_PATH due to special topology arrangements. However, one cannot count on these paths to be accepted by other routers, leading to potential routing blackholes. Thus, to improve routing robustness, it is best to exclude such routes from BGP decision process if alternate routes exist.

2. Missing mandatory attributes. Another BGP semantics violation is missing mandatory attributes in the routing updates. Some attributes such as ORIGIN or AS_PATH cannot be easily inferred. However, the NEXT_HOP attribute, which is also mandatory, usually is the interface IP address of the advertising router. Proactively correcting this prevents unnecessary session resets.

3. Private information. For eBGP sessions *i.e.,* BGP sessions between routers belonging to different ASes, BGP attributes in general should not contain private information such as private IP addresses/prefixes or private AS numbers. For iBGP sessions, *i.e.,* sessions within an AS, such values are meaningful only within the local network. For all sessions, bogon prefixes or unallocated prefixes by address registries should not be announced. Accidentally accepting such routes may impact forwarding for legitimate destinations using private address blocks inside the local network.

Remark: In general, there is only a small classes of updates that violate BGP semantics due to the flexibility of routing policies and ambiguities of the protocol specification. If the RouteNormalizer is initialized with the *local routing policies*, more semantic violation can be identified. Routing updates in this category can be corrected or filtered using route filters. However, there is a limited number of filters a router can accommodate, and this imposes additional overhead.

2.3 Fix violations of routing policies

Although individual network providers have the freedom to define their own routing policies, there are some well-known guidelines for specifying policies according to the best common practices (BCP) of BGP [15]. Violations of policies may result in unexpected traffic blackholes.

1. Export policy violations. The RouteNormalizer identifies the class of updates that violate routing policies, especially those associated with the local AS. For example, a multi-homed customer, or a customer peering with more than one upstream providers is not allowed to advertise routes received from one provider to another [18]. Such violations of the so-called export policies can be identified by checking the AS relationship between the customer AS and the nexthop AS beyond the customer AS in the AS_PATH of the routes advertised by the customer router. ISP can either deprefer these routes or resort to overlay routing to bypass the problem.

2. Nexthop violations. Typically the routes advertised by a neighboring router in the BGP session correspond to the routes in the forwarding table of the neighbor. The nexthop AS and the nexthop IP should be the neighbor's AS number and the remote router's interface IP respectively. Otherwise, the routes advertised will not correspond to traffic going through the neighbors. If any of the two assertions fails, the RouteNormalizer raises an alarm to the network operator.

Remark: Routing policy violation checks are more cumbersome to perform inside routers due to the need of external information such as AS relationships.

2.4 Detect routing anomalies

The RouteNormalizer identifies routing anomalies by examining the routing data locally received from the neighbors. Correlating updates from multiple locations can provide a network-wide view to help consistency checking and potentially discover additional routing anomalies. Network researchers have used BGP data from multiple vantage points to improve AS relationship inference [18, 41, 8] and BGP health monitoring [32]. Our goal here is to (i) identify deviations from average behavior, (ii) perform consistency checking, and (iii) track announcements of one's own address blocks. Note that timely and accurate detection of routing anomalies facilitate and inform mitigate responses.

1. Anomalous routing behavior. The RouteNormalizer establishes a routing profile consisting of characteristics such as the distribution of routes in terms of AS_PATHs, ASes, and the number of routes from each neighbor. This profile is tracked over time and across data from each vantage point. Besides deviation from history data, another way to

find anomalies is to identify frequently changing values, implying instability in the routing system.

2. Routing inconsistency. The second objective is to identify inconsistent route advertisements excluding convergence effects. Commonly each AS advertises its best path consistently across all peering locations to all its neighbors complying to its export policies. This assumption is commonly held to be true according to the protocol specification, also based on the well-known definition of an AS [22]. The consequence of inconsistent routing advertisements is unintended routing behavior deviating from the usual practice of hot-potato routing, resulting in potential increased network cost.

3. Address space hijacking. This is an important anomaly given today's danger of IP address hijacks and traffic blackholes. This also illustrates the usefulness of correlating BGP data from multiple locations. Address space hijacking refers to the case when a network announces a route as the originator to the address block it does not own. The last AS in the AS_PATH is the originating AS. Spammers are known to take advantage of hijacked address spaces to avoid being identified. Such instances have also occurred due to misconfigurations: in December 1999, AT&T Worldnet was off the air because someone by mistake was advertising a critical network owned by Worldnet [9]. To detect hijacking attempts for *locally originated address blocks*, the RouteNormalizer uses the knowledge of which address blocks originate from the local network. It is important to identify whether other networks announce as the originator updates to locally owned address blocks. To increase the confidence in detection, we correlate the suspected hijacked address blocks with other data sources, such as Spam Archive [5] and blacklisted addresses from sites such as Dshield [1].

Remark: Anomaly detection is not supported by routers today and is difficult to implement in routers due to the complex logic and external data requirement. Commercial routers used in most networks today are not programmable. Implementing anomaly detection in software on RouteNormalizer is more flexible and easily allows additions of new functionality as the protocol evolves.

2.5 Manage instability and load

Routing instability and attacks can incur numerous updates, as demonstrated by worm outbreaks causing routing disruptions [16]. Processing such updates adds extra overhead given router's limited resources. The RouteNormalizer helps manage the load on routers by mitigating routing instability and minimizing unnecessary updates processed by the local router in the following ways.

1. Identical routing updates. We found on average about 5% of updates from RouteViews [6] consist of identical BGP updates which is usually due to router software bugs [28]. These updates are not at all useful for route computation; however, they may consume router resources. The RouteNormalizer can easily detect their presence and drop them.

2. Instability due to flapping prefixes and session reset. Occasionally, a large number of routing updates stem from unstable prefixes that continuously go up and down [38] due to flaky hardware, for instance. Route flap damping [44, 36] requires maintaining the update history for each prefix and can lead to memory exhaustion. Furthermore, it may not be enabled in every router. By delaying routing updates, the RouteNormalizer is effectively slowing down the sending rate, needed to prevent router overload. The RouteNormalizer can also effectively emulate the flap damping algorithm in a modified and improved way by ensuring that routes are only suppressed when at least one alternate route exists.

Receiving a large number of legitimate updates from its neighbors due to sudden significant routing changes or session resets will causes significant update processing overhead. Usually the core router has multiple BGP sessions, which can be easily overwhelmed and become unresponsive in forwarding packets [13]. The *graceful restart mechanism* for BGP [39] has been proposed to minimize the effects due to session resets. To take advantage of this feature, routers need to support such capability so that End-of-RIB marker is sent and routes are retained even after session reset for a bounded time. However, many routers today may not support such capability, especially the legacy routers with outdated router software. The RouteNormalizer can emulate the graceful restart functionality and furthermore enhance it by ensuring there are no inconsistent routing information.

Whenever the RouteNormalizer delays routing updates on behalf of the local router, the imposed delay can be set based on the inferred load. Inference is performed by observing the sending rate of routing traffic and data traffic if available. Moreover, unstable routes may affect routing decisions if such routes are preferred over alternate stable ones.

Remark: Functionality provided by the RouteNormalizer to deal with routing instability cannot be easily implemented inside the router, as doing so would directly impact the router load. Precisely when routers are overloaded, such functionality is critical in preventing the impact on the forwarding plane.

2.6 Detailed normalization algorithms

We have enumerated the main functionalities of the RouteNormalizer. Here we describe some of them in more details, focusing on their benefit and improvement over the equivalent functionalities in the router. In all the following cases, the router either does not provide such support or our algorithm significantly improves upon it.

Deaggregation detection: Deaggregation, the opposite of aggregation, refers to the behavior of advertising many small prefixes covered in larger prefixes already present. The negative consequence is that the router receiving such announcements may experience memory exhaustion, possibly leading to router crashes. To protect against deaggregation, routers currently use the *Max-Prefix Limit* [14] feature, which by default disables the peering session after the number of received prefixes exceeds the configured maximum number [25]. However, the router does not attempt to differentiate between regular and deaggregated prefixes, consequently causing the entire BGP session to be affected.

The RouteNormalizer more intelligently deals with prefix deaggregation, which can be easily detected by observing an increase in the number of prefixes while the number IP addresses remains relatively constant. When router memory is scarce, routing announcements to prefixes which are contained within existing prefixes in the routing table can be safely dropped without impacting reachability. It may impact routing decisions given differences in routes between the aggregate and the subnet prefix.

Address hijacking detection: This functionality is currently not supported by routers and will be difficult for routers to provide due to the complex logic and external data requirement. Detecting address hijacking relies on having accurate prefix to origin AS mappings; however, there are no such authoritative data sources available. If we generate an alarm for each update that indicates a different origin AS from the latest route of the prefix, there would be many false positives. The reason is that due to multi-homing there are legitimate reasons for Multiple origin ASes (MOAS [47]). To remedy this, we develop a mapping of prefix to origin AS by learning from history data from multiple vantage points to improve detection accuracy.

Graceful restart: Some routers today support graceful restart [39] and assume that within a configurable time limit the restarting router can still properly forward traffic. The RouteNormalizer can emulate this and enhance routing consistency if it can observe data traffic. The key is to observe whether traffic such as TCP ACK packets are arriving from the remote router indicating that packets can indeed reach the destinations. Otherwise, the RouteNormalizer will withdraw the routes advertised by the remote router for which alternate routes exist at the local router to ensure traffic is not blackholed unnecessarily. Note that even if due to asymmetric routing, no return traffic is observed, reachability is not compromised as only alternate routes are chosen.

Before the session is re-established, the RouteNormalizer keeps track of the latest updates from the local router to the restarting remote router. Once the session comes up, the remote router reannounces its entire forwarding table to the RouteNormalizer, which in turn only selectively forwards routes that were previously withdrawn and any changed routes compared to those before the session reset. From the RouteNormalizer to the remote router, the latest local router's forwarding table is sent. The added intelligence ensures that only the necessary routes are exchanged upon session reestablishment to reduce overhead for the local router.

Instability detection: BGP already has route flap damping as specified in RFC2439 to deal with routing instability. The RouteNormalizer can emulate it if it is not supported by the local router or disabled due to memory usage concerns. This would help reduce both processing and memory overhead. The RouteNormalizer further improves it by handling persisent flapping, which is ignored due to reinitialized penalty values upon session reset. Damping statistics are remembered after session reset to detect such routing instability.

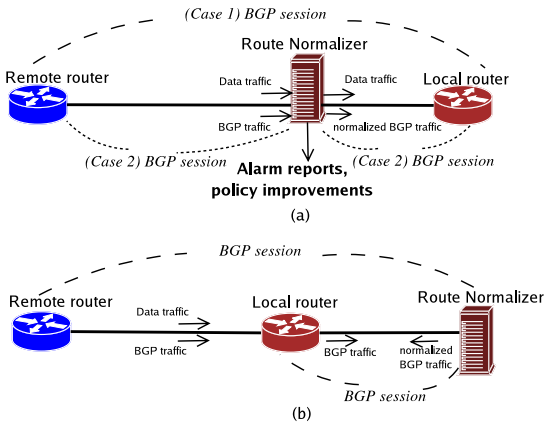

Figure 2. RouteNormalizer (RN) single router deployment scenario.

3 Deployment Scenarios

In this section, we illustrate deployment scenarios with different degrees of benefit in terms of functionality and ease of deployment. We expect both eBGP and iBGP to benefit from deploying the RouteNormalizer to block routes from untrusted external networks as well as to prevent misconfigurations from propagating across the internal network.

Figure 2 depicts how the RouteNormalizer is used for a single BGP session protecting the local router from one remote router. We expect the RouteNormalizer to be deployed very close to the local router. There are two main ways of setting it up, distinguished by whether the RouteNormalizer can observe data traffic. Shown in Figure 2(a), case 1 is the transparent TCP proxy setup, requiring no configuration changes of existing BGP sessions. The RouteNormalizer intercepts any packets between the remote and the local router. It inserts, modifies, and drops any packet destined to the BGP port. The presence of the RouteNormalizer is completely transparent to either router. Case 2 of Figure 2(a) illustrates the approach where the RouteNormalizer establishes two sessions, with the remote and local router respectively. Remote router needs no configuration changes as it treats the RouteNormalizer as the local router, which is made aware of the RouteNormalizer. Changes to local routers are usually easier to implement. To address the shortcoming of the first setup, one can adopt the approach shown in Figure 2(b). The RouteNormalizer can pretend to be local router from the perspective of the remote router, whose configuration requires no modifications. The local router is configured to have a BGP session with the RouteNormalizer to receive normalized routes. Note that the local router forwards the BGP updates between the remote router and the RouteNormalizer which does not observe data traffic to other destinations within the local network. The resulting advantage is that it can be implemented as a software-based router and does not need to forward high-speed data traffic. This setting is more appropriate for BGP sessions in the core Internet with high traffic rate.

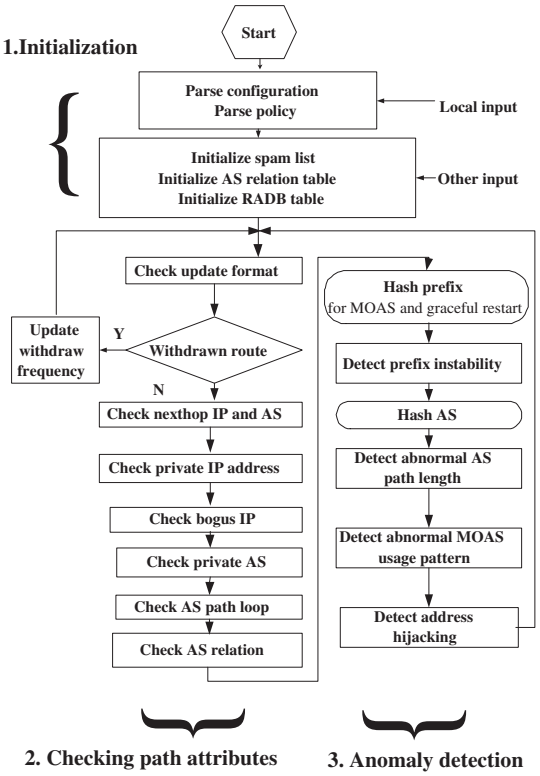

Figure 3. Functionality implementation.

In general, an IP network consists of many BGP routers, each of which may peer with multiple routers. To reduce the management overhead of per router neighbor based deployment, we generalize Figure 2(a) Case 2 to protecting multiple "local" routers, where the RouteNormalizer peers with multiple local and remote routers, which is equivalent to the deployment setup of Routing Control Platform (RCP) [12].

4 Prototype Evaluation

We have implemented a prototype of the RouteNormalizer with all the functionalities described in Section 2 with the exception of protection against resource-based router DoS attacks, which is our future work. We evaluate the performance of the prototyped RouteNormalizer based on the deployment setup with two separate BGP sessions shown in Figure 2(a) Case 2. The performance of other deployment settings is similar and not included here due to the space constraint.

4.1 Functionality implementation

The RouteNormalizer receives update messages and processes them according to the dataflow shown in Figure 3.

1. Initialization. There are three sets of initialization input files. The first are local router's configuration files, containing useful information such as locally announced address blocks and import filters. The second are user-defined policy configurations for the RouteNormalizer. This policy configuration is kept secret to prevent attackers from evading the RouteNormalizer even given its source code. We note that even using the default configurations it is nontrivial to evade our anomaly detection techniques. The third is external information to improve RouteNormalizer's confidence in generating accurate alarms, for example, the Routing Assets Database (RADB) and Spam Archives [5].

2. Checking path attributes. After initialization, the RouteNormalizer performs normalization actions on BGP update attributes as described in Section 2. The order in which the checks are performed is determined by impact severity starting with the most serious violations. Processing for a given update is stopped if a violation is detected and cannot be corrected. The RouteNormalizer first checks for update format errors by removing unknown attributes. It updates the withdrawal frequency for the corresponding prefix.

For announcements, the RouteNormalizer first corrects if needed the nexthop IP and AS number to match the advertising router. It subsequently checks if the announced prefixes contain private addresses or unallocated addresses. Then it performs private AS number checks, loop detection, and AS relationship violation checks in succession. Note that checking AS relationship violation is the most time consuming part because of searching the relationship for each consecutive AS pair in the AS path. This consumes 70% of total processing time. It subsequently performs anomaly detection on attribute values to find deviations from history.

3. Anomaly detection. The RouteNormalizer uses past history to perform anomaly detection. The use of history is justified as history provides information on usable routes. It first detects prefix related anomalies followed by AS path related anomalies. This includes detecting unstable routes, anomalous attributes such as unusually long AS paths, significant changes in the number of prefixes announced by an AS, and abnormal origin patterns to infer address hijacking attempts. To facilitate anomaly detection, the RouteNormalizer stores relevant state for received update messages in two hash table data structures as shown in Figure 3.

Note that we choose the hash table data structure instead of Patricia tree because we need to keep track of all distinct prefixes even if one is covered by another for remembering different routing attributes. Upon receiving an BGP update, the RouteNormalizer updates the corresponding records in both hash tables. Our prototype is an extensible framework as each functionality is implemented as an independent module.

4.2 System performance evaluation

The RouteNormalizer is implemented at user level written in C with about 3,400 lines of code. Our prototype testbed is shown in Figure 4, where we use a Cisco 3600 with IOS 12.2(26a) as the local router. Since we do not focus on the routing performance on the remote router, we use GNU Zebra v0.94 [2] based software router running on a PC as the remote router. The RouteNormalizer is evaluated using a Dell Dimension 8400 with 3GHz Pentium 4 Processor and 1.5GB memory running on Linux Fedora Core 3. It establishes two BGP sessions: one with the local router, the other with the remote router. To study the routing behavior of the local router, we set up a peering session to a peering router

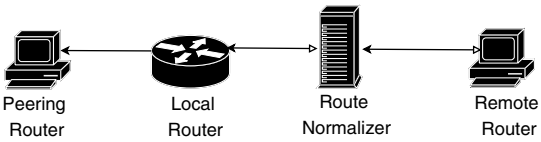

Figure 4. RouteNormalizer prototype testbed.

using another zebra instance running on a PC. All links are full-duplex 100Mbps switched Ethernet.

1. System throughput. We examine the overhead of route normalization in handling high volumes of routing traffic by modifying Zebra [2] (called *pseudo-Zebra*) to read update messages from files and send them out in the format defined in the RFC 1771 [37] as fast as possible over the network to overcome the minimum route advertisement timer constraint of Zebra software router and achieve the maximum throughput over 100 Mbps link. We observe that the average throughput using our pseudo-Zebra program is 77.9Mbps or 64,916 packets/sec on the testbed, which is comparable to the Bro traffic normalizer [23]. Note that this throughput result is obtained when the router is reading from files. The sending rate is thus limited by the file I/O on the remote router.

Handling multiple peers has only slight degradation on the throughput. We argue that this throughput is acceptable because the data rate of BGP update traffic is significantly lower than 77.9Mbps due to minimum route advertisement timer and the router processing overhead, as confirmed using empirical BGP data. For example, the peak rate of BGP updates for about 30 peers is less than 80Kbps, much less than the maximum traffic rate the RouteNormalizer can sustain. It takes on average only 223 seconds for the RouteNormalizer to process a single day's routing update data for 36 peers, assuming the data is readily available. Thus, we expect the RouteNormalizer can effortlessly keep up with the BGP update traffic rate in real time.

2. Memory consumption. The memory consumption for storing both *PrefixHash* and *ASHash* increases linearly during the initialization stage. It subsequently remains quite stable, increasing very slowly when processing new updates. For example, keeping states for 16 days of routing messages from a single peer consumes less than 20MB of memory. To ensure memory does not grow without limit and to prevent state exhaustion attacks, we use the strategy similar to LRU cache replacement policies by timing out memory usage. The amount of memory consumed increases linearly but very slowly with increasing number of peering sessions. With 30 peers, the memory consumed is slightly less than 150MB. The average amount of memory used per peer is 5MB, much less than the 20MB for a single peer because of the information shared among peers.

5 Empirical Evaluation using BGP Data

In this section, we evaluate the effectiveness of the RouteNormalizer using empirical data from public BGP source – RouteViews [6].We study the experimental findings of BGP updates that can benefit from route normalization.

Category	# updates (% total alarms)	# ASes involved	# prefixes involved
AS path loops	1,047 (3.5%)	23	2,483
Private ASes in AS paths	930 (3.1%)	31	953
Unusually long AS paths	172(0.57%)	1305	256
AS relationship violations	20,174 (67%)	438	94
MOAS violations	5,976 (19.86%)	382	267
Unstable prefixes	1,785 (5.9%)	58	1496

Table 2. Identified normalizable updates (RouteViews: October 2006)

Our analysis is not meant to be exhaustive and the results reported here focus on one month of data in October 2006 using three months of data from July 2006 to September 2006 as history information for anomaly detection. From the routing related email complaints on the North America Network Operator Group (NANOG) mailing list [3], the RouteNormalizer can identify most of them, confirming its effectiveness at identifying real routing problems.

5.1 Normalization statistics

Table 2 shows the overall number of identified updates in each category and number of ASes and prefixes involved. We notice that the AS relationship violations appear to constitute the majority of identified alarms. Altogether there are $30,110$ alarms generated during the one month time period using RouteViews data from 36 BGP routers. Table 2 also shows the number of ASes involved in each category: 438 ASes responsible for AS loops. We did not find any NEXT_HOP violations or instances of bogon prefixes in the BGP feeds examined. For some categories such as "unusually long AS paths", we define a threshold based on observed data distribution in history data. Although we were unable to detect updates in all the categories implemented, *e.g.,* hijacked address blocks, partly due to lacking local configuration information, our current findings are encouraging.

To justify the use of history data for detecting anomalies in the BGP routing attributes, we analyzed the distribution in the routing information across each prefix. We found that on average 75% of prefixes have only fewer than 12 distinct routes over the three months history data. Focusing on only the AS path and Origin attributes, the two most common attributes directly impacting routing decisions, on average 94% have at most 5 distinct routes. These statistics show that history is a good predictor for identifying routing anomalies, as the routing attributes are fairly stable over time.

To generate concise alarm reports in real time, related alarms are grouped together to produce aggregated alarm reports. In this prototype, we use a simple and intuitive technique to group the alarms based on the time of occurrence, the ASes and prefixes involved in the alarms. In our analysis, we use 5 minutes as the threshold for maximum separation across alarms, as typically routing convergence occurs within minutes. Furthermore, we use 10 minutes as a limit for aggregating a long running alarm, as operators would like to be notified of routing events in real time. The threshold values are set by observing the distribution of alarm intervals.

By grouping related alarms across different prefixes, we reduced the number of alarms from 635 to 221 by 66% on average per peer by examining 10 peers for 10 days. Grouping together different yet closely-occurring alarms for the same prefix helps identify problems associated with the same destination. We further reduced the number of alarms to 128, *i.e.,* by 43% on average. Finally, we experimented with grouping based on the network impacted by the alarm, *i.e.,* the affected AS. This results in 96 alarms on average, ranging from 36 to 118 with the standard deviation of 62, a reduction of 25%.

5.2 Case study: graceful restart

To support the graceful restart functionality and improve routing consistency, the RouteNormalizer needs to record the latest route for each prefix. This imposes extra overhead during the first time the BGP session is established. We evaluate this overhead by measuring the time for transferring the entire routing table with 144,615 entries (taken from RouteView data for AS7018 on October 1 2006) from the remote router to the local router, and then propagating to the local router's peer shown in Figure 4. The local router is Cisco 3600 with IOS 12.2(26a). Without the RouteNormalizer, the transfer takes 194.23 seconds on average with standard deviation of 3.7. With the RouteNormalizer, the duration for table transfer is increased to 201.5 seconds on average with standard deviation of 5.7. The extra delay imposed is around 7.27 seconds, which we believe is acceptable as it is one-time overhead and per-prefix penalty is very small.

To evaluate the benefit of the enhanced graceful restart on the RouteNormalizer, we use BGP data from RouteViews on October 1 2006 and set the local router to be multihomed to the AT&T (AS7018) and Sprint (AS1239) network, two tier-1 ISPs. Route selection algorithm is based on the BGP RFC [37]. The benefit of graceful restart are two fold: reducing both the number of updates exchanged and the duration that routes are potentially unusable. In our analysis, we artificially bring down the session to AS7018. Among 144,615 routing table entries, only 68,066 routes need to be withdrawn. Assuming the session is down for 90 seconds, 9 new updates are sent after session reestabilishment in addition to 68,062 withdrawn routes, compared to 144,683 updates if no graceful restart is available. The duration for transferring such updates is only 90.6 seconds with the RouteNormalizer, as opposed to 194 seconds in the case of transferring the entire table to the local router without the RouteNormalizer. Note that the saving results from the fact that the 76,621 stale routes are still in local router's routing table and do not need to be sent to the local router upon session re-establishment and to be propagated to the peering router. Thus these routes are usable as soon as the link recovers. This translates to reducing the average time duration that the route is potentially unusable at local router by 23.96 seconds for each prefix.

5.3 Known routing problems

The NANOG mailing list [4] regularly reports ongoing problems with the Internet routing system, as they may impact user performance and network operations. Using simple keyword search and manual inspection of the emails, we identified 54 events that are clearly related to routing problems during the time period from October 2001 to November 2006. Part of these real routing problems are caused by either malicious intent or misconfigurations [17]. The RouteNormalizer will detect these problems on the wire and attempt to mitigate against the negative impact.

Table 3 shows the overall statistics on detecting the routing problems reported on the NANOG mailing list. For a given reported routing problem, RouteViews routing data spanning the time period starting from two days before the event until the end of the day of the report are analyzed. In most categories, the detection percentage is quite high. Note that some of these routing problems may be related to the internal networks and thus not visible in BGP updates or may not be observable at the BGP feeds we have access to.

Private addresses: On June 16, 2002 at around 15:43PM, Qwest network leaked routes for 10.0.0.0/8. The RouteNormalizer detects this and raises an alarm. Depending on its policy, it may drop such updates.

Prefix leaking which violates AS relationships: On July 11, 2003, one network operator complained that traffic originated from Sprint arrived over ALGX (AS2828) customer's interface, violating export policies. The RouteNormalizer reports the AS relationship violation for path 1239 6395 14751 2828 2828 8001. Broadwing Communications (6395) did not filter the announcement from its customer (AS14751), which incorrectly readvertised routes learned from its provider AS2828 to its other provider AS6395. If alternate paths exist for the destination prefix, this route should not be used.

Prefix deaggregation: On May 2, 2002, within 15 minutes an additional 5,000 routes originated from AS705 entered the global view. The RouteNormalizer detected this by observing a sudden increase in the number of prefixes advertised by AS705; however, there is no increase in the number actual IP addresses. Most of these newly announced prefixes are /24s and were contained in larger /8 and /11 address blocks. If permitted by the policy, the RouteNormalizer can simply drop these announcements to ensure that the local router's routing tables are not overwhelmed.

AS loop: The RouteNormalizer found 12 events of AS-level routing loops in BGP data but there are no corresponding email complaints posted on NANOG mailing list.

Address hijacking: On November 9 2006, 86 prefixes such as 12.0.0.0/7, 121.0.0.0/8, 15.0.0.0/8 were announced by AS29449 instead of its usual origin ASes such as AS7018, lasting for over 10 minutes. The reason of this event is still unknown. Using three months of history data, the RouteNormalizer detects this by observing that these prefixes never used AS29449 as its origin AS. Consequently, the RouteNormalizer raises suspected address hijacking alarms.

Instability: On October 5, 2005, Level3 Communications Inc. (AS3356) terminated its peering relation with Cogent Communications Inc. (AS174). This depeering event causes

Category	Private address	AS relationship violation	Prefix deaggregation	Address hijacking	Routing instability
Normalizer functionality	Private IP address detection	Export policy violation detection	Routing instability mitigation	Anomaly detection in attributes	Routing instability mitigation
Number of detected events	6	5	7	2	15
Detection ratio (total)	100%(6)	83%(6)	70%(10)	28.6%(7)	60%(25)

Table 3. Identified NANOG routing related problems (RouteViews: Oct 2004–Nov 2006).

reachability problems in many locations on the Internet, mostly for customers who are singly homed to Level3 or Cogent. Using Level3's AS3356 BGP feed, the RouteNormalizer observes explicit withdrawals for 98.7% or 2,936 distinct prefixes which originally had an AS path containing AS174. Among these withdrawn prefixes, 1,063 distinct prefixes were originated from AS174, accounting for all such prefixes. Examining the number of distinct IP addresses instead of IP prefixes, we detect withdrawals of 99.3% or 26,482,692 IPs going through AS174, and 100% or 20,541,927 IPs originated from AS174. This occurred over the time period of 289 minutes. Using the data from RouteView's Cogent AS174 feed, the RouteNormalizer detects similar routing dynamics. The RouteNormalzer also detects the restoration of the peering two days later on October 7, 2005. All prefixes previously withdrawn were re-announced. Identifying events impacting a large number of prefixes, the RouteNormalizer marks these routes as anomalous. If the local router has alternate routes for these destinations, the RouteNormalizer will conservatively suggest to continue using the alternate routes even when these anomalous routes are re-announced due to instability concerns.

6 Security Considerations

As a firewall for routers, the RouteNormalizer can be attacked in various ways, *e.g.,* resource overload attacks mentioned in traffic normalizer [23]. Our emulation of the graceful restart feature is one such example. Malicious peers can continuously reset the BGP session. Our strategy is to assign a penalty value per peer related to session reset to identify repeatedly flapping sessions. The penalty increases with each session reset and decreases over time, similar to route flap damping [44]. Another example is that attackers may focus on the most time-consuming functionalities such as checking for AS relationship violation by generating updates with abnormally long AS paths. The RouteNormalizer needs to search the relationship between each consecutive AS pairs in the AS path. Receiving many such paths may result in decreasing throughput. Our strategy is to calculate the actual AS path length ignoring AS prepending and set a threshold for the AS path length. Whenever a peer sends a significantly large number of updates with long AS paths, we raise an alarm. Moreover, we can optimize the relationship checking by storing examined AS path in the memory to reduce the number of comparisons. Another optimization is to focus on the relationship violations that directly impact the local AS, reducing the number of checks needed. For protection, the RouteNormalizer is numbered with private addresses, so that external users cannot directly send packets to it.

7 Related Work

It has been well-known that the Internet routing is vulnerable to various misconfigurations and attacks [11]. Recent studies [47, 29, 42] have focused on identifying routing anomalies using BGP data. Several protocol enhancements [40, 35, 42, 19] have been proposed to secure routing protocols. However, they are either incomplete or requires modification of the routing protocol leading to slow adoption. Complementary approaches exist without modifying BGP to identify configuration errors. IRV [20] defines such a service to mitigate malicious or faulty routing information by relying on collaboration among several networks. Feamster *et al.* [17] applied statical analysis to find faults in BGP configurations. Caesar *et al.* [12] proposed a centralized routing control platform (RCP) to facilitate configuration and route selection inside an AS. Karlin *et al.* [26] proposed an enhancement to BGP to slow the propagation of anomalous routes, similar to our design. Karpilovsky and Rexford [27] recently proposed an algorithm to reduce router memory usage by discarding alternate routes and refreshing on demand. Our work mainly differs from these work in that the RouteNormalizer operates online and actively identifies and *correct* routing updates, as well as influence routing decisions without affecting reachability. RouteNormalizer can directly use the RCP platform to take advantage of data from multiple vantage points for performing route normalization.

Parallel to our approach, instead of normalizing routing updates, Handley *et al.* [23] designed a data traffic normalizer for network intrusion detection. Protocol scrubbers [31, 46] have also been proposed for removing network attacks at both transport and application layers. Wang *et al.* [45] proposed shield – a lightweight network filter in end-system to protect against known vulnerability.

8 Conclusion

In this paper, we presented the detailed design of the RouteNormalizer which helps protect against external routing misbehavior observed at a local router by identifying and correcting malicious and misconfigured routing updates. The deployment of such a platform requires little to no changes in router configurations and no protocol modifications. Using a prototype implementation evaluated in a commercial router testbed, we showed that it can achieve good performance and scalability to support current BGP traffic rate. More importantly, we validated the benefit of routing anomaly detection in real time by analyzing empirical BGP data. The results from correlating with routing related complaints on the NANOG mailing list indicate that the RouteNormalizer can

identify most of the known routing events impacting user performance.

References

[1] Distributed Intrusion Detection System. http://www.dshield.org/.

[2] GNU Zebra – routing software. http://www.zebra.org.

[3] NANOG Mailing List Information. http://www.nanog.org/mailinglist.html.

[4] NANOG Maling List Information. http://www.nanog.org/mailinglist.html.

[5] Spam Archive, Donate Your Spam to Science. http://www.spamarchive.org/.

[6] University of Oregon Route Views Archive Project. www.routeviews.org.

[7] Cisco Security Advisory: Cisco IOS BGP Attribute Corruption Vulnerability, 2001.

[8] G. Battista, M. Patrignani, and M. Pizzonia. Computing the Types of the Relationships Between Autonomous Systems. In *Proc. IEEE INFOCOM*, March 2003.

[9] S. Bellovin. Where the Wild Things are: BGP Threats. Nanog 28 Talk, Jun 2003.

[10] V. J. Bono. 7007 Explanation and Apology. NANOG 97-04.

[11] K. Butler, T. Farley, P. McDaniel, and J. Rexfod. A Survey of BGP Security Issues and Solutions. Technical Report Technical Report TD-5UGJ33, AT&T Labs - Research, 2004.

[12] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and Implementation of a Routing Control Platform. In *Proc. 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[13] D. Chang, R. Govindan, and J. Heidemann. An Empirical Study of Router Response to Large BGP Routing Table Load. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2002.

[14] S. Chavali, V. Radoaca, M. Miri, L. Fang, and S. Hares. draft-chavali-bgp-prefixlimit-02.txt: Peer Prefix Limits Exchange in BGP, April 2004.

[15] S. Convery and M. Franz. BGP Vulnerability Testing: Separating Fact from FUD. Nanog 28, June 2003.

[16] J. Cowie, A. T. Ogielski, B. Premore, and Y. Yuan. Internet Worms and Global Routing Instabilities. In *Proc. SPIE*, 2002.

[17] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[18] L. Gao. On Inferring Autonomous System Relationships in the Internet. In *Proc. IEEE Global Internet Symposium*, 2000.

[19] V. Gill, J. Heasly, and D. Meyer. The BGP TTL Security Hack. NANOG 0302.

[20] G. Goodell, W. Aiello, T. Griffin, J. Ioanmidis, P. McDaniel, and A. Rubin. Working around BGP: an Incremental Approach to Improving Security and Accuracy in Interdomain Routing. In *Proc. NDSS*, 2003.

[21] T. G. Griffin and G. Wilfong. On the Correctness of iBGP Configuration. In *Proc. ACM SIGCOMM*, August 2002.

[22] S. Halabi and D. McPherson. *Internet Routing Architectures*. Cisco Press, Indianapolis, Indiana, second edition, 2000.

[23] M. Handley, C. Kreibich, and V. Paxson. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proc. USENIX Security Symposium*, 2001.

[24] J. W. S. III. *BGP4 Inter-Domain Routing in the Internet*. Addison-Wesley, 1999.

[25] C. S. Inc. BGP Restart Session After Max-Prefix Limit. .

[26] J. Karlin, S. Forrest, and J. Rexford. Pretty Good BGP: Improving BGP by cautiously adopting routes. In *Proc. International Conference on Network Protocols*, 2006.

[27] E. Karpilovsky and J. Rexford. Using forgetful routing to control BGP table size. In *Proc. CoNext*, 2006.

[28] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet Routing Convergence. In *Proc. ACM SIGCOMM*, 2000.

[29] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfigurations. In *Proc. ACM SIGCOMM*, August 2002.

[30] F. Majstor. Attack on the Routing Protocols. RSA 2002. Cisco System Inc.

[31] G. R. Malan, D. Watson, F. Jahanian, and P. Howell. Transport and Application Protocol Scrubbing. In *Proc. IEEE INFOCOM*, 2000.

[32] D. McGrath. Passive Internet Health Monitoring With BGP. Nanog 0310.

[33] D. Meyer and A. Partan. S-BGP/soBGP Panel: What Do We Really Need and How Do We Architect a Compromise to Get It? NANOG 0306, June 2003.

[34] S. Murphy. BGP Vulnerabilities Analysis. IETF draft June 2003.

[35] J. Ng. Extensions to BGP to Support Secure Origin BGP (soBGP). IETF Draft: draft-ng-sobgp-bgp-extensions-01.txt, November 2002.

[36] C. Panigl, J. Schmitz, P. Smith, and C. Vistoli. RIPE Routing-WG Recommendations for Coordinated Route-flap Damping Parameters, October 2001. Document ID: ripe-229.

[37] Y. Rekhter and T. Li. A Border Gateway Protocol. RFC 1771, March 1995.

[38] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP Routing Stability of Popular Destinations. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[39] S. R. Sangli, Y. Rekhter, R. Fernando, J. G. Scudder, and E. Chen. Graceful Restart Mechanism for BGP. IETF Internet Draft, June 2004.

[40] Stephen Kent and Charles Lynn and Karen Seo. Secure Border Gateway Protocol (Secure-BGP). *IEEE J. Selected Areas in Communications*, 2000.

[41] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proc. IEEE INFOCOM*, 2002.

[42] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proc. first Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[43] US-CERT. Multiple Denial-of-Service Vulnerabilities in Cisco IOS.

[44] C. Villamizar, R. Chandra, and R. Govindan. BGP Route Flap Damping. RFC 2439, 1998.

[45] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits. In *Proc. ACM SIGCOMM*, 2004.

[46] D. Watson, M. Smart, G. R. Malan, and F. Jahanian. Protocol Scrubbing: Network Security Through Transparent Flow Modification. *IEEE/ACM Transactions on Networking*, 2004.

[47] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. An Analysis of BGP Multiple Origin AS (MOAS) Conflicts. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, November 2001.