A Blueprint for Improving the Robustness of Internet Routing

Georgos Siganos, Michalis Faloutsos

Abstract— **Protecting BGP routing from errors and malice is one of the next big challenges for Internet routing. Several approaches have been proposed that attempt to capture and block routing anomalies in a proactive way. In practice, the difficulty of deploying such approaches limits their usefulness. We take a different approach: we start by requiring a solution that can be *easily* implemented *now*. With this goal in mind, we consider ourselves situated at an AS, and ask the question: how can I detect erroneous or even suspicious routing behavior? We respond by developing a systematic methodology and a tool to identify such updates by utilizing existing public and local information. Specifically, we process and use the allocation records from RIR, the local policy of the AS, and records used to generate filters from IRR. Using our approach, we can automatically detect routing leaks, and routing deviations from the registered policy of the AS. To the best of our knowledge, this is the first fully automated approach that can achieve this. Additionally, we identify some simple organizational and procedural issues that would significantly improve the usefulness of the information of the registries. Finally, we propose an initial set of rules with which an ISP can react to routing problems in a way that is systematic, and thus, could be automated.**

I. INTRODUCTION

The Border Gateway protocol (BGP) [26] sits at the heart of Internet routing, and is inevitably facing many security and robustness problems. The first problem is the unauthorized advertisement of IP prefixes. For example, in 1997, AS7007 [19] de-aggregated and advertised a large portion of the Internet, attracting traffic away from its rightful owners, thus creating a 'black-hole' for Internet traffic. The second type of problem is the use of illegitimate paths [22]. The traffic is going to the right destination but over the wrong path. The wrong path here means that the traffic goes through ASes that should not and do not want to carry it. These problems can appear either because of malice or human error, to which BGP is especially vulnerable [18]. Part of the problem is that configuring the routers is complicated, the available tools are usually low-level with no static correctness checking, and no immediate feedback control on possible errors. As a result, it is difficult to predict what will happen with a configuration change [9] and trial-and-error is often used.

BGP has evolved in an incremental way [13][29][6][10] and has partially addressed some of these security requirements. But, there is a need for more security [21][20][7]. Several *proactive* approaches have been proposed [17], [16][15][12][14][28][30], and IETF has a established a working group [3] to investigate and recommend routing security requirements. The most well-known and advanced approaches are S-BGP [17], [16] proposed by BBN, and SoBGP [15] proposed by CISCO. A quick overview of the various approaches is presented in the next section and in a recent survey [24].

G. Siganos and M. Faloutsos are with the Dept. of Computer Science & Engineering,University of California, Riverside, Email:{siganos,michalis}@cs.ucr.edu

The current proactive proposals have several problems. First, most of the solutions require significant changes in the routing protocol. Second, many solutions are computationally intensive requiring significant processing and resources at the router, which many current routers may not be able to provide [16]. Third, many approaches require additional global infrastructure such as certification authorities. Fourth, these solutions have limited usefulness when partially deployed. Thus, the benefit of the first deployments is minimal, and no one is willing to make the start. Fifth, ISPs are commercial entities that are driven by profitability and customer demand, and proactive solutions can be quite expensive to deploy. Last, but not least, many proposals focus more on the technical and engineering aspects, and less on the usability and user friendliness. However, network operators are reluctant to adopt complex and difficult to manage solutions.

We take a different approach: we require a solution to be *easy* to implement *today*. Thus, we want our solution to work: (a) with the existing information, (b) without additional large-scale infrastructure, (c) without requiring global participation and conformance, (d) in a lightweight and cost effective manner. We believe that a feasible solution that can satisfy these requirements is a reactive approach. We place ourselves in the position of a network administrator and we consider two different sources of BGP updates:(a) *local* updates, which the AS receives from its direct neighbors, and (b) *remote* updates from cooperating routing collectors or observation points[1]. Given a BGP update (prefix, AS path) we want to identify if it is 'valid'. We want to check if the origin AS, last AS in the path, is really the owner of the prefix, and if the AS path is legitimate. More specifically, we have three tasks: for the *local* updates, (a) we validate the origin to avoid using and propagating routing leaks[2], for the *remote* updates, (b) we validate the origin of prefixes we own to detect hijacking of our IP space, and (c) we validate the legitimacy of the paths that include our AS number to detect routing exploitation of our AS policy. Furthermore, the above validation tasks can be performed by a single AS or several cooperating ASes in

[1]Note that the distant monitoring information is important since in some cases it is easier to detect misbehavior. For example, if an AS hijacks the space of ASA, the erroneous advertisements may never reach ASA as local updates.

[2]Validating the paths for the local updates requires the knowledge of the policies of many ASes, and thus is currently unfeasible.

a distributed way. These ASes could form a group of trust, exchange information, and look out for violations of each other policies, in a similar fashion as the Neighborhood Watch program in real life. The motto for the Neighborhood Watch program is "We look out for each other" and this can also be applied in the Internet case.

Our main contribution is that we develop an approach to validate the origin AS and path of a BGP update from the point of view of an AS. Our approach is intended to act as an advisor to a network administrator. In addition, our approach could be the first step towards an automated reaction approach, if and to the extent that the administrator feels comfortable with. In more detail, our work consists of three components: (a) we develop a validation methodology and a tool, (b) we apply our method to real data, and (c) we propose ways to improve the capability of ASes to respond to routing errors.

A. A new methodology. We rely on *currently available information*, mainly the public Internet registries: RIR for validating the origin, and select information from IRR for validating the paths. We process and clean the information in order to minimize the effect of inaccurate information. Second, we follow an *AS-centric view* and make an effort to rely on local information as much as possible.

B. A real world study. We apply our methodology on real updates collected by three different routing collectors for two separate time periods. First, we analyze a time period of 13 days to assess the effectiveness of our methodology and approach. Second, we analyze a time period of 8 days to study a major routing leak. Our goal is to: (a) examine the feasibility of the method, (b) assess the quality of the available information, and (c) profile the behavior of Internet routing. In our study, we find that:

- Our method is feasible and effective: For the origin AS validation, we can validate approximately 97% of the prefixes. When we evaluate our reactive approach, the number of suspicious updates is quite small, usually less than 1 to 3 per hour. For the path validation, there exist over $4,000$ ASes, that we could not find any path deviation.
- Many ASes are quite unprepared and slow to respond to route leaks. We study a major routing leak, and we find that it took ASes over one hour to respond and erroneous updates were circulating 6 days later.

C. Improving Internet's ability to react. Our work suggests two major avenues for improving the ability to react to routing problems. First, we provide an assessment of the current state of the registries, and provide simple suggestions and practices that could improve the usefulness of the information. Second, we develop a systematic approach that an AS can use for reacting to misbehavior. Our proposal is only a first step towards a systematic and potentially automated reaction scheme.

The rest of this paper is structured as follows. In section II we present some definitions and background work. In section III, we describe our framework. In section IV, we examine our framework with real data. In section V, we present the profile of a major routing leak. In section VI we discuss the necessary steps to make our approach even more effective. In section VII we present our conclusions.

II. BACKGROUND AND PREVIOUS WORK

In this section, we briefly describe an overview of Internet routing, the routing registries, and solutions for the secure operation of Internet.

A. *Internet and BGP-4*

Internet is structured into a number of routing domains that have independent administrations, called **Autonomous Systems (AS)**. Each autonomous system is identified by a number, **asn**, which is assigned to it by an Internet registry. An Autonomous System uses an intra-domain routing protocol, like OSPF or IS-IS, inside its domain, and an inter-domain protocol to exchange routing information with other Autonomous Systems. The defacto standard for inter-domain routing is **BGP-4** [26]. The primary difference between the intra-domain and the inter-domain protocol is that the first one is optimized for performance, solely based on operational requirements, while the second is used to enforce the **policy** of the Autonomous System, which corresponds to the **business relations** with its neighboring ASes.

An Autonomous System given its policy, will advertise to its neighbors a list of **IP Prefixes** that are reachable through it. Each route is tagged with a number of **attributes**. The most important attribute is the **AS_PATH**. The AS_PATH is the list of ASes that packets towards that route will traverse. An AS uses **filters** to describe what it will import from and export to a neighboring AS. The filter can include a list of prefixes, a list of regular expressions on the AS_PATH, a list of communities, or any possible combination of these three.

B. *Resource Allocations: Regional Internet Registries(RIR)*

Administrative procedures are necessary to ensure the uniqueness of the IP addresses and Autonomous System numbers. The registration process is coordinated by the Internet Assigned Numbers Authority **IANA**. The registration is happening in a hierarchical fashion, in which IANA allocates parts of the Internet address space to regional Internet registries **RIR**. Currently, there are four RIR established[3]. **ARIN** serving North America, a portion of the Caribbean, and sub-equatorial Africa. **RIPE** is serving Europe, the Middle East, Central Asia and African countries north of the equator. **APNIC** is serving the Asian Pacific region and **LACNIC** is serving the south America. RIR subsequently allocate IP space to National IR **NIR** or directly to Local IRs **LIR** usually large ISPs, which in their turn **allocate** resources to the **end users**, corporations and other ISPs. The community supports the RIR by paying annual fees based on how many resources they consume. For example for RIPE, the annual fee for an extra small organization is €1,750, while for an extra large organization the fee is €6,500.

The LIR and end users of the IP allocations are required to utilize the address space in an efficient manner. They need to

[3]A new RIR, Afrinic, was officially established in 2005 and is responsible for parts of Africa. In the time period we examined the records that are now part of Afrinic were part of either RIPE or ARIN.

```
// RPSL Format
inetnum:      213.68.0.0 - 213.71.255.255
status:       ALLOCATED PA
mnt-by:       RIPE-NCC-HM-MNT
mnt-lower:    UUNETDE-I
mnt-routes:   AS1270-MNT

inetnum:      213.70.90.80 - 213.70.90.95
status:       ASSIGNED PA
mnt-by:       UUNETDE-I

//SWIP Format
NetHandle:           NET-216-160-0-0-1
OrgID:               USW
NetRange:            216.160.0.0 - 216.161.255.255
NetType:             allocation
TechHandle:          ZU24-ARIN
```

Fig. 1. Example of (partial) Allocation records for RPSL and SWIP.

maintain detailed documentation to justify every **assignment** of resources. For example, in ARIN region, an ISP should have documented every assignment that contain eight or more addresses. The RIR may, at any time, ask for this information. If the information is not available, future allocations may be impacted or current allocations may be taken back. The basic criteria that should be met to receive prefixes are a 25% immediate utilization rate and a 50% utilization rate within 1 year. Additionally, in order to request a new allocation, an ISP must show at least 80% utilization of its current allocation. The assignments within an allocation are checked routinely for correctness when a LIR requests for a new allocation. For example, RIPE will make 3 random checks of assignments and will ask documentation to evaluate them.

The previous part describes the current procedures for allocation of IP space. Internet has evolved in both the IP address architecture and the administrative procedures used. First, Internet moved from a classful address to a classless address architecture. During this first period resources were allocated using classes and were provided very liberally to organizations with minimum requirements. The were five classes used. In a class A allocation, the first 8 bits were used to identify the network, while the remaining 24 to identify the end host. A number of organizations have selfishly maintained these allocations. We refer to these allocations as **LEGACY**. In a class B and C allocation, the first 16 and 24 bits identify the network while the last 16 and 8 bits the host. We refer to these early allocations as **ERX-RIR**. The remaining two classes were used for multicasting and for experimental use, and we don't use them in our analysis. Note, that for both LEGACY and ERX-RIR, we refer to them as separate RIR even though technically they are not. The reason is that even if their records physically exist in the four main RIR, the RIR have no authority on these records.

The RIR use a number of different formats to register the allocation records. RIPE and APNIC use **Routing Policy Specification Language (RPSL)** [5] [8], while ARIN uses SWIP [25], and LACNIC use a mix of RPSL and SWIP. The NIR that exist in the APNIC region seem to use an RPSL based format. In Figure 1, we have an example of allocation records for prefixes in both RPSL and SWIP. Note that these are partial records. The first is the allocation record for $213.68.0.0/14$. Note, that the maintainer of the record is a RIPE maintainer and that it allows maintainer UUNETDE-I, to register further assignments. An example of such an assignment is $213.70.90.80/28$. For the SWIP case, we have the OrgID attribute that can help us find the hierarchy in assignments and the correlation with the AS numbers.

```
as-set:     AS-5
members:    AS5, AS5:AS-CUSTOMERS
mnt-by:     AS5-MNT

as-set:     AS5:AS-CUSTOMERS
members:    AS2
mnt-by:     AS5-MNT

route:      199.237.0.0/16
origin:     AS5
mnt-by:     AS5-MNT


aut-num:    AS5
import:     from AS6 action pref = 100; accept ANY
import:     from AS4 action pref = 90;
                accept <^AS4+ AS4:AS-CUSTOMERS*$>
import:     from AS2 action pref = 80;  accept AS2
export:     to AS6 announce AS-5
export:     to AS4 announce AS-5
export:     to AS2 announce ANY
mnt-by:     AS5-MNT
```

Fig. 2. Example of (partial) policy records of an AS.

RIR Dataset: For our analysis, we use the registries of December 28, 2004. The registries contain $3,417,553$ prefix allocations and $31,105$ AS number allocations and $2,277,091$ technical personnel contacts. Note, that in addition we analyze the registries of January 09, 2005 to capture the change in the registration records in that time period[4]. We should stress here that our evaluation is based on public data[5].

C. Routing Policy: Internet Routing Registries(IRR)

The need for cooperation between Autonomous Systems is fulfilled today by the Internet Routing Registries (**IRR**) [1]. The main uses of the IRR registries are to provide an easy way for consistent configuration of filters, and a mean to facilitate the debugging of Internet routing problems. ASes use the RPSL to describe their routing policy. At present, there exist 70 registries, which form a global database to obtain a view of the global routing policy. Some of these registries are regional, like RIPE or APNIC[6], other registries describe the policies of an Autonomous System and its customers, for example, cable and wireless CW or LEVEL3.

The design goal of RPSL is twofold. First, RPSL provides a standard, vendor independent language, so that the policy of an AS can be published in an easy to understand format. Second, RPSL provides high level structures for a more convenient and compact policy specification. There exist 12 different types of records, that either describe portion of a policy, or describe who is administering this policy. In Figure 2, we have an example of partial policy RPSL records for an Autonomous

[4]With the exception of ARIN, since we were given access only to the December 28, 2004 records

[5]ARIN and LACNIC require an AUP agreement prior to providing access to their bulk whois data.

[6]Note that both RIPE and APNIC have a single registry for both allocation records and policy records. ARIN, maintains a separate registry for the policy but is not widely used and LACNIC maintains no registry for policy.

System. The **route** class is used to register the IP prefixes an AS can originate. The **as-set** and **route-set** classes are high level structures that can be used to group prefixes. For example an AS can create an as-set that will contain the prefixes of its customers. Finally, the aut-num class contains the import and the export policies for every neighbor of the AS. Note that every class has a mnt-by attribute that specifies the maintainer of the record. This is done for security reasons so that only the maintainer can update that record.

In our previous work [27], we have developed a methodology to analyze the registered policy. Our tool Nemecis can solve problems such as merging multiple registries and cleaning the registered policy. Additionally, we can reverse engineer the policy of an Autonomous System, check for possible errors and find the correlation between the import and export rules. This way we can check the consistency of the registered policies.

D. Proposed solutions

Next, we present in more detail two of the most popular proactive solutions S-BGP and SoBGP.

S-BGP:S-BGP has three main security mechanisms. First, they use Public Key Infrastructure (PKI) to authenticate every aspect of a routing message, like the ownership of an IP address, the ownership of the Autonomous System (AS) that originates that prefix, the identity of the AS, and so on. Second, they introduce a new optional transitive attribute to carry digital signatures. A router can use the signatures and the information in PKI to validate the BGP update information. Third, they are using IPsec for point-to-point security. The main characteristic is the number of digital signatures required to verify an update. They need to verify every AS in the path, and the origin AS. Additionally, when an AS sends an update to its neighbors, their approach needs to calculate a new one for every neighbor that it has. In their approach they can use cache to improve performance, but when we have changes in either the topology, or in the policy they need to recompute them. They mention that for a router with 30 peers, they can achieve 9 operations per second, with no cryptographic hardware installed in the router.

SoBGP: The SoBGP approach has two main goals. First, to validate the authorization of an AS to advertise an IP prefix. Second, to prove that there exist at least one valid path to the destination. SoBGP is using three certificates to sign information. The certificates are the Entity certificate, which is used to authenticate the identity of an autonomous system. This certificate can be signed by any organization that the receiver trusts. The authorization certificate ties an AS to the IP prefixes that is allowed to advertise. This certificate can be signed by the organization that delegated the IP prefix to the AS. Last, they have the policy certificate, which describes the policies related to the IP prefixes and the topology of the advertising AS. This certificate is signed using the private key of the AS. The keys and certificates, form a web of trust, and no central repository of keys exist, or PKI infrastructure like the one S-BGP requires. The certificates are per Autonomous System, and every AS is responsible for

Algorithm 1 $validate\_originAS(prefix, asn)$

```
1:  $inetnums \leftarrow find\_prefix\_allocations(prefix)$
2:  $routes \leftarrow find\_routes\_with\_origin(prefix, asn)$
3:  for $inetnum$ in $inetnums$ do
4:      $org\_inetnum \leftarrow find\_prefix\_organization(inetnum)$
5:      for $route$ in $routes$ do
6:          $org\_route \leftarrow find\_route\_organization(route)$
7:          if $org\_inetnum$ == $org\_route$ then
8:              return strongly validated
9:  for $inetnum$ in $inetnums$ do
10:     $org\_inetnum \leftarrow find\_prefix\_organization(inetnum)$
11:     $org\_ases \leftarrow find\_organization\_ases(org\_inetnum)$
12:     if $asn$ in $org\_ases$ then
13:         return strongly validated
14: if $routes$ not $empty$ then
15:     return weakly validated
16: return not validated
```

storing its own database of keys and certificates. In order to prove that the path exist, they are building a directed graph using the information stored in the policy certificates, and check if the path is feasible. A key component is that they don't require any encryption processing to be done on the routers. The processing can either be done on the routers or it can be done by servers in an offline manner.

III. Framework for security

In this section, we present our framework and show how we can validate a BGP announcement. First, we start with how we validate the origin AS, and then we show how we can validate the path.

Data for origin Validation: For the origin AS validation, we use mainly the allocation records of RIR. Our framework uses the fact that RIR allocate to an organization prefixes and AS numbers independently. Thus, any AS number that an organization handles can be the origin AS of the prefixes it administers. Note, that using the route record, an organization can identify which of these ASes can be the actual origin AS.

Data for path Validation: For the path validation, our framework can check the validity of the path from a point of view of an AS, without requiring the knowledge of the policy of the other ASes in the path, which frequently are regarded as business confidential. Thus, every AS can independently check for the legitimacy of the paths that contain its AS number. For the validation, we use from IRR the definition of the sets that this AS is using to generate its filters. This is a requirement for peering between ASes in many regions of the world: ASes need to keep their sets up-to-date otherwise valid announcements will be blocked.

A. Origin AS Validation

The first priority is to check the validity of the origin AS. What we try to find is: Given the prefix I and the corresponding path $P = [a_1, ..., a_{i-1}, a_i]$, check that a_i can be the origin of I. The main algorithm is Algorithm 1. First, in lines 1 and 2, we find all records that contain the prefix both for the allocation records and route records that register the AS as the origin AS. Next, in lines 3 to 8, we check if

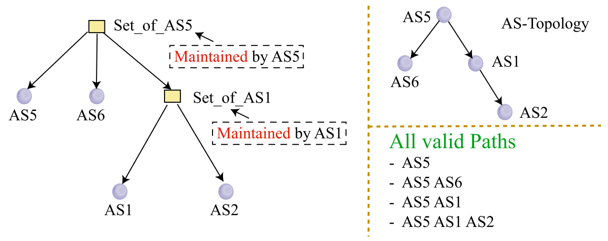

Fig. 3. An example of how a set is transformed to valid paths.

a prefix allocation record and a route record are maintained by the same organisation. This works mainly for the RIPE and APNIC registries, because ARIN has a very small IRR registry and LACNIC has none[7]. Next, in lines 9 to 13, we check if we can find that the origin AS and the prefix are part of the same organization. The first two cases are **strongly** validated, because the information to correlate the prefix and the origin are maintained by the same organization and are tied to the allocation records (RIR). If we can not find the necessary information using the RIR records, we will use the route records in IRR. These cases are **weakly** validated, because any AS can register that it is the origin of the prefix. We run the algorithm for both a_i and a_{i-1} in the case the a_i can not be strongly validated. The reason we check both ASes is to capture cases such as the provider has the prefix allocations but the prefixes are used by its customer.

Depending on the goal we want to achieve, there can be many different modes of operation for the origin AS validation. For example, if we want to detect malicious users, then the validation should only use the strongly validated cases. A malicious user can simply register a new route object in one of the IRR registries and thus avoid detection. In this paper, we focus more on how to detect misconfigurations and human errors, and thus we can use more relaxed criteria.

In this spirit, we also use a number of empirically derived rules for the validation. We refer to them as **empirical rules** and can be grouped in two categories. In the first category, we use common information between already validated (origin AS, prefix) tuples and tuples we want to validate. In the second category, we use the references to technical personnel to correlate between prefixes and AS numbers. Regarding the first category, we can validate an origin tuple if either we have validated a less specific prefix with the same origin AS. Additionally, if a validated tuple and a not validated share the same DNS server and the same origin AS then we can also validate the tuple. Regarding the second category, if we have the same technical contact for a prefix allocation and an AS number then the origin tuple is valid. Additionally, if for the contact information associated with the prefix allocation and the AS number, the email server is the same, the origin tuple is considered valid. For the remaining origin AS, which are not decidable validated, if we don't have conflicting data we assume that the origin tuple is valid. The conflict arises if we have another prefix that includes or is including the prefix we examine, and has a different origin AS. In our evaluation we didn't find an empirical rule that we use much more frequently than the others.

[7]Note, that the route records are part of an IRR registry. The difference between an IRR registry that is run by a RIR is that there exist consistency checks so that an organization can register a route record only if it is authorized via the allocation record.

Algorithm 2 $set\_to\_path\_check(sets, subPath)$

```
1:  $currentSets \leftarrow sets$
2:  for $asn$ in $subPath$ do
3:      $found \leftarrow false$
4:      $newSets \leftarrow []$
5:      for $member$ in $currentSets$ do
6:          if $member$ is set and $asn$ maintains $member$ then
7:              $found \leftarrow true$
8:              $Push(newSets, member)$
9:          else if $asn$ == $member$ then
10:             $found \leftarrow true$
11:     if $found$ is $false$ then
12:         return $not\ valid$
13:     if $newSets$ is not $empty$ then
14:         $currentSets \leftarrow newSets$
15: return $valid$
```

B. Path Validation

We develop an approach to check the validity of a path from the point of view of an AS. The question we answer is: Given the policy of an ASA and a path $P = [..., b_2, b_1, A, a_1, a_2, ...]$ is this a valid path that conforms to the registered policy of ASA? Note, that AS A cuts the path P into two **sub-paths**, $P_{before} = [b_1, b_2, ...]$ and $P_{after} = [a_1, a_2, ...]$. Thus, the validation of the path P can be broken into two separate parts, the validation of P_{before} and the validation of P_{after}.

First, we focus on how we can validate P_{before} . The policy of ASA is dictated by what the AS imports and exports to its neighbors. The import specifies what paths or prefixes AS A will accept, and export specifies the prefixes that ASA will act as a transit AS. To analyze the policy, we need to match the import and exports rules. For this task, we use our tool Nemecis [27]. For the P_{before} case, we are interested in the prefixes that ASA imports from b_1 and subsequently exports to a_1. These prefixes will help us validate the sub-path P_{before}. A straightforward validation approach would be to find all the prefixes the origin AS in the sub-path can originate and then check if the policy of every AS in the sub-path allows these prefixes to be imported and then exported to the following AS. However, we don't need to use such an approach, because it has a number of disadvantages like requiring the policies of many ASes. What we use instead is the high level structures that the RPSL based policy offers.

Let us describe our approach in more detail. In RPSL, as we mention in section II, instead of using directly the prefixes to describe the import and export filters, it is common to use the abstraction of AS numbers and as-sets. An example of how to use an as-set to find all the possible valid sub-paths is shown in Figure 3. Assume that we have AS5 that maintains the set, Set_of_AS5 and that AS5 has two customers, AS1 and AS6. AS5 can use the abstraction of AS number and sets to register the filters instead of manually adding the prefixes. So in the case of AS1, AS5 can use the set that AS1 maintains.

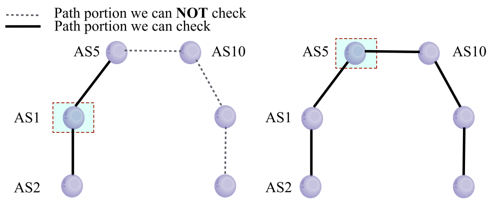

Fig. 4. Policy centric path validation. We can validate the full path only when it goes through peers and customers.

If AS5 exports its set Set_of_AS5 to a neighbor AS, then the hierarchy built in the set, mandates that possible valid sub-paths should be like [AS5,AS1]. The details of the algorithm on how to validate a sub-path using a set can be found in Algorithm 2.

Next, we need to validate P_{after}. The straightforward way will be to check the policy of every AS in the sub-path. Our approach works as follows. We consider that we have the reverse path, that is, we consider that the last AS in the sub-path is the origin AS. This way, we can use the same procedure we described above for the P_{before}. This approach is based on the assumption that whatever ASA registers as import from a_1, is what a_1 exports to A. Even though this might not always be the case if the policy of the AS is not updated, these cases will generate path flags.

Conceptually, our approach can be used to check the full path using the policy of a single AS if the path goes through its customers and peers. If the path goes through a provider, we can check the path up to that AS. In Figure 4, we have a visualization of this. The link between AS5 and AS10 is a peer to peer link, while for the remaining of the links it is provider to customer. In this example, if we check the path from the point of view of AS1, then we can only check the path up to AS5. On the other hand, if we check the path from the point of view of AS5, we can check the whole path. In the first case AS1 will import ANY from its provider AS5 and will subsequently export ANY to its customer AS2, thus any path that goes through AS5 is valid for AS1. We should stress here, that in our analysis we don't use the abstraction of business relations. We use the business relations only for illustration purposes to provide a better intuition and motivation of our approach. Our approach relies solely on the import and export filters.

In later sections, we will use the term **origin tuple** to refer to the (origin AS, prefix) tuple, and **path tuple** to describe the (AS,Path) tuple for a prefix. Additionally, we refer to the tuples that we can not validate as **flags**.

IV. BGP Validation

In this section, we start with the origin AS validation, then we analyze the policy validation for the AS path. In both cases, we first analyze in general how well the validation works, and then we investigate how a reactive scheme works.

For our evaluation, we analyze the BGP routing tables and updates during a time period of 13 days, starting at December 28 2004. We analyze three collectors, *rv2*(routeviews2) [23] in North America (USA), *rrc03* [2] in Europe (Holland), and *rrc06* [2] in Asia (Japan). In table I, we have a summary of the routing collectors. The largest in terms of peers is rrc03 with 79 AS peers and 108 peerings. RV2 is the largest in terms of routing entries and the number of updates it received. The rrc06 collector is much smaller than the previous two, but we analyze it for geographical diversity.

TABLE I
Route Collectors **Data** Summary

Collector	rrc03	rv2	rrc06
Peers(AS/Total)	$79/108$	$34/40$	$6/6$
Routing Table	$2,887,967$	$5,739,807$	$153,491$
Updates	$36,658,783$	$72,549,959$	$2,558,233$

TABLE II
Route Collectors **Origin Validation** Summary

Collector	rrc03	rv2	rrc06
Unique (Prefix,AS)	$164,152$	$177,507$	$158,498$
Number of Flags	$6,008$	$6,109$	$6,039$
Percentage of Flags	3.6%	3.4%	3.8%

A. Origin AS validation

The first validation is to check whether the origin AS is authorized to advertise the prefix associated with the path. We examine all unique origin tuples, i.e. (prefix, origin AS), found in a collector. We find the origin tuples by analyzing both the routing table, which is our starting point, and the BGP updates that the collector received during the 13 days period. In table II, we show the number of unique origin tuples and the percentage of origin tuples that raised a flag in our approach. For rrc03 we have $6,008$ flags out of the total $164,152$ origin tuples. The percentage of flags is 3.6% for rrc03, 3.4% for rv2 and 3.8% for rrc06. This result is both positive and negative. On the positive side, we have over $158,000$ origin tuples that can be validated. On the negative side, the allocation records *should* be accurate. In the next part, we examine in more detail how we validate the origin tuples and analyze the flags per RIR and per origin AS. Our analysis focus on the rrc03 collector.

Examining prefixes per RIR: We start by classifying every prefix, found in the origin tuples, according to its RIR. We have $64,272$ prefixes from the ARIN region, $29,071$ from RIPE, $29,242$ from APNIC, $7,483$ from LACNIC, $29,661$ from ERX-RIR and $4,423$ from LEGACY. Note, that old allocations, ERX-RIR and LEGACY, have a significant number of prefixes.

ARIN and ERX-RIR have significant contribution in the number of flags. In Figure 5, we have the number of flags per RIR. We have $3,000$ flags for ARIN but only 232 flags for RIPE. The contribution of ARIN and ERX-RIR, is significantly larger than the contribution of the others. In table III, first, we have the percentage of flags compared to the total number of origin tuples for every RIR. For ARIN we have 4.7%, which is an order of magnitude larger compared to RIPE, APNIC and LACNIC. Additionally, the old allocations have a comparably high percentage of flags with 5.7% for ERX-RIR and 8.7% for the LEGACY IP space. Second, we

TABLE III
PERCENTAGE OF FLAGS PER RIR(RRC03)

	ARIN	RIPE	APNIC	LACNIC	ERX-RIR	LEGACY
Per RIR	4.7%	0.79%	2.1%	1.1%	5.7%	8.7%
All Flags	49.9%	3.8%	10.2%	1.4%	28.2%	6.4%

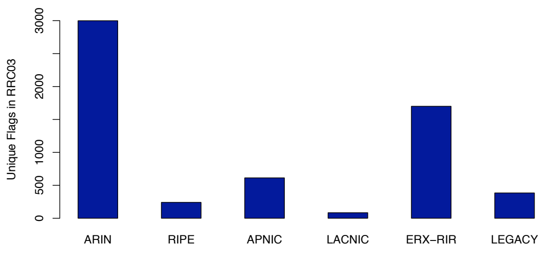

Fig. 5. Unique (Prefix,AS) origin tuples that can not be validated per RIR.

have the percentage of contribution for the total number of flags. 50% of the flags are from ARIN while 28% are from ERX-RIR. These results reveal that potentially there exists a problem with the ARIN registry and with old allocations.

RIPE is the best maintained RIR. In Figure 6, we analyze the different RIR by examining how we validate the origin tuples. As we describe in section III, we have three categories, the strongly validated, the weakly validated and the empirical rules. The best overall RIR is RIPE where most of the origin tuples can be validated in a strong way. A surprising result is that APNIC is not performing as well as RIPE even though they use the same registry format. If we compute the percentages, we have that 73% are strongly validated for RIPE but only 40% for APNIC, while we have 51% for ARIN and 61% for LACNIC. In APNIC, we can validate more origin tuples using route records than allocation records. One possible explanation for this poor performance is the existence of national registries(NIR) within the APNIC region. We will talk in more details about this problem in section VI.

Examining flags per origin AS: Next, we look for patterns of flags by correlating the flags by the origin AS. In Figure 7, we plot the flags for an AS versus the total prefixes this AS originates. As we see from the figure, we have flags both from ASes that originate a small number of prefixes and from ASes with a large number of prefixes. This shows that there are no implicit patterns. For example, we don't have the case that only large ASes generate flags. Thus we need our system, since we can not focus only on a few ASes.

US administered ASes are the main source of flags. Next, in Figure 8, we find for every origin AS that creates a flag, the country of registration. The vast majority of ASes that create flags are caused by US administered ASes. Note, that this holds across administration areas. For example, most of the flags within the RIPE region are caused by US administered ASes. Another interesting point here is that the second column is for ASes from Turkey. Most of these flags are due to a single AS that advertised erroneously prefixes from all RIR areas. This is the known event of AS9121 [4] which advertized over 100,000 prefixes to its peers. What is not known is that even though the event happened in December 24 2004 and believed to have lasted for a day, we could see its effect on December

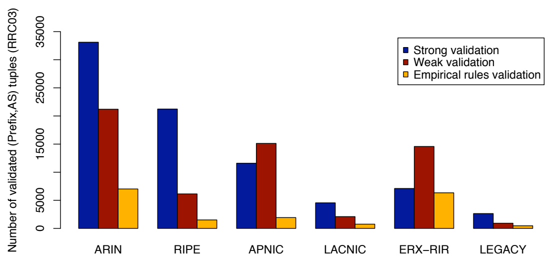

Fig. 6. Details on how we validate the origin AS.

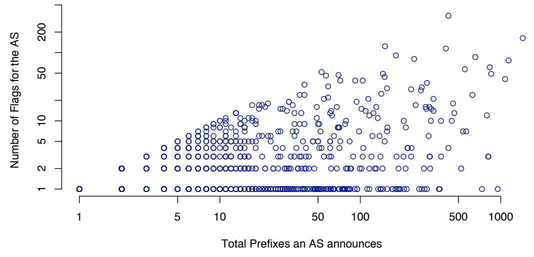

Fig. 7. Distribution of number of unvalidated prefixes an AS originates.

28 and for at least two more days for a small number of prefixes. We will examine that event in section V. The fourth spot, Unknown, is for ASes that we could find no allocation records in any RIR region.

B. Reactive origin AS Validation

We will consider a fictitious case where we would like to validate events as they arrive. We would only need to check unique events, which then we could cache and remember. With our system, we assume that we will only need to check the flagged events, thus our scheme can act as an administrator advisor. We find that usually one would need to check no more than one flags per hour.

We start with the origin tuples found in the routing table of December 28, 2004. We take these tuples as given, and we examine the updates for the next 13 days. We try to validate every unique origin tuple that we see for the first time, which we refer to as **event**.

Caching alone does not help much. How many new events do we see over time? If we keep seeing the same updates the need for our tool may be limited. Simple caching of legitimate events would eventually ensure that we only accept good updates. In Figure 9, we plot the number of unique events and the corresponding events that caused a flag versus the 13 days of observation, we aggregated the time in intervals of one hour for visualization purposes. We find that it is not uncommon to have a large number of events like over 100 events per hour, and it can go as high as 500 events[8]. Additionally, we don't find any reduction in the number of events as time progresses. This shows that a scheme that is solely based on history to validate the prefixes would not help in practice, since there are too many new origin tuples to validate. When we use the information stored in RIR, thus find the flags, we see that we have much fewer events to investigate.

[8]We have cropped the y axis to 180 max, the points at 36 go to 538 and at 298 go to 235 flags.

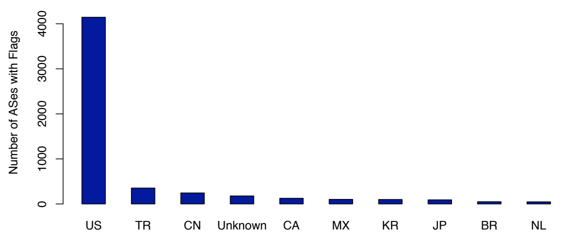

Fig. 8. The number of flags for the top 10 countries.

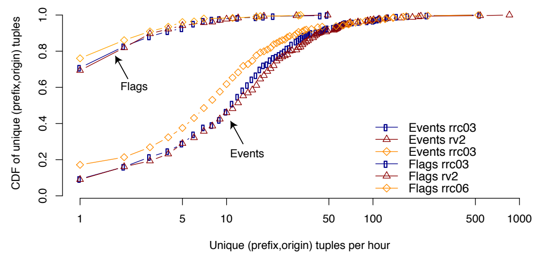

Fig. 10. CDF of unique events and flags per hour for various collectors.

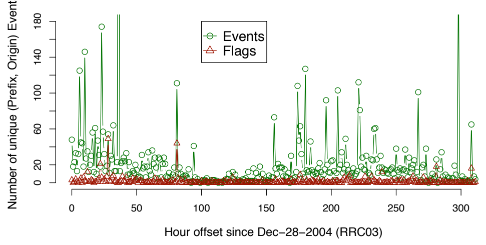

Fig. 9. The evolution of unique (prefix,origin AS) origin tuples

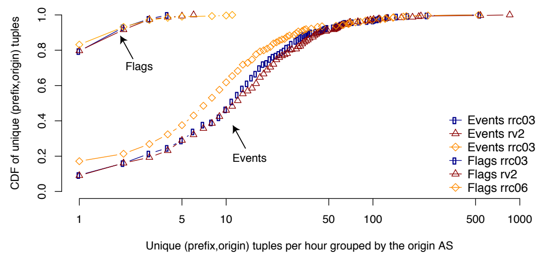

Fig. 11. CDF of unique events and flags per hour grouped by the origin AS for various collectors.

We usually have 0 to 3 flags per hour. We examine in more detail the flags per hour. In Figure 10 we plot the Cumulative Distribution Function (CDF) for the unique events and events that caused flags for all three collectors. Note that the x axis is in log scale. As we can see the total events for the three collectors follow the same pattern, and the rrc03 and rv2 collectors have an almost identical distribution. We have a 50% probability to have more than 10 events per hour, and a 10% probability to have over 40. On the other hand for the flags, the probability to have equal or less than 0 and 1 flags in an hour for rrc03 is 51% and 70% respectively. The probability of having less or equal than 3 flags in an hour is 88%. Additionally, we have a maximum of 48 flags per hour. This can be potentially a problem, but as we show next the flags are not independent of each other and thus we can minimize even further the cases that need to be investigated.

AS-based correlation of events and flags. In Figure 11, we plot the number of total events and flags aggregated in intervals of one hour and grouped by the origin AS within that interval. This means that if for example AS1 was the origin AS for three flags during a time period of one hour, we have only one AS-based flag and not three. The reason we do this is that usually these flags are correlated to one incident and the administrator can analyze them as one. Using this approach, we find that we have a maximum of 4 AS-based flags per hour for rrc03, 6 for rv2 and 11 for rrc06. Additionally, with 79% probability we have equal or less than 1 AS-based flags per hour for rrc03. These results are quite encouraging and show that we can achieve a significant reduction in the number of flags.

Next, we focus on rrc03 and we classify the flags based on their RIR. In Figure 12, we plot the CDF of AS-based flags per hour. As expected, ARIN and ERX-RIR, have a much higher probability of having flags. Note, that for RIPE and LACNIC, the probability of having zero AS-based flags per hour is around or over 95%, while APNIC follows with 90%.

Duration of flagged origin tuples. The next question is for how long flags are present in a routing table. If the flags are present only for a small time period then the ability or even the need to a reaction could be limited. We focus on rrc03 and RIPE, APNIC and LACNIC that have the fewest percentage of flags, and thus it is more likely that these flags could be actual routing leaks. First, we want to investigate how **persistent** these flags where, that is after they appeared compute the percentage of time these flags were present in the routing table. Note, that the withdraw of a flag can be both explicit, via a withdraw, or implicit if the peer advertises for this prefix a new path. In Figure 13, we plot the histogram of the persistence of the flags. The persistence of the flags is bimodal. First, we have the flags that are present for a small percentage of time, usually they last for less than 30% from the time we first saw the announcement. We have around 300 such cases, 30% of the total flags. The remaining 70% of the flags were very persistent in the sense that they were visible until the end of our data from the time they appeared.

In Figure 14, we plot the histogram of the duration in hours of flags that have a persistence equal or less that 30%. We find that flags can last for over 40 hours sometimes close to 90 hours. On the other hand, we have around 23 flags that last for less than one hour. Note, that that the flags that lasted approximately 40 hours, where actual leaks as we will see in the next section.

To summarize the origin AS validation results, we show that a reactive approach could be effective but mainly against misconfigurations and human errors. Even if we include all prefixes, the number of flags is sufficient low to guarantee a low overhead in validating the origin AS. Problems exist mostly in the ARIN region and old allocations, but these are not so serious as to prevent the effective deployment of a reactive approach.

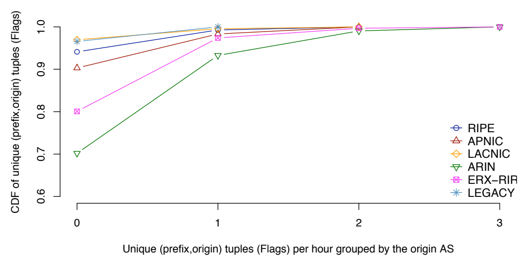

Fig. 12. CDF of unique events that raised flags, grouped by the origin AS for various collectors.

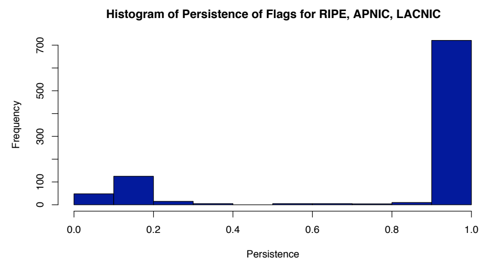

Fig. 13. Persistence of flags for the three RIR (RIPE,APNIC,LACNIC).

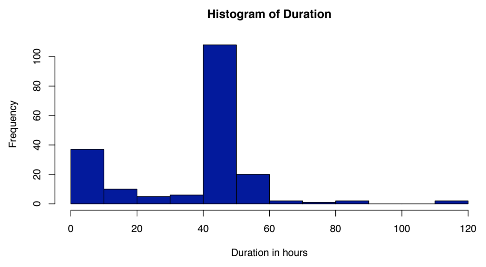

Fig. 14. How long the unstable flags last, rounded to hours.

TABLE IV
ROUTE COLLECTORS **PATH VALIDATION** SUMMARY

Collector	rrc03	routeviews2	rrc06
All ASes	$19,022$	$19,142$	$18,887$
Unique Paths	$963,935$	$1,959,951$	$51,743$
All ASes with Policy	$10,313$	$10,377$	$10,429$
#of path tuples (AS,Path)	$3,754,159$	$7,366,628$	$189,924$
Valid tuples	$1,684,391$	$2,494,471$	$88,040\%$
Percentage Validated	44.8%	33.8%	46.3%
ASes with Good Policy	$8,322$	$8,273$	$8,582$
#of path tuples (AS,Path)	$1,597,545$	$2,158,422$	$83,742$
Valid tuples	$1,268,243$	$1,626,246$	$78,002$
Percentage Validated	79%	75%	93%

C. Path validation

The second type of validation is to check whether an advertised AS path is valid. We validate a path from the point of view of an AS in the path. It is reasonable to expect that the AS that wants to validates its paths knows its policy, and thus can easily detect all deviations. In our case, since we are not the administrator of an AS, we use the IRR registries to check the paths for a number of ASes. We try to understand under the current conditions what we can detect, and how useful and effective our approach can be. This section is more on evaluating the registered policies. We only check the path for ASes that have a reasonably updated policy. There are two parameters we require: First, we need to be able to understand the registered policy and match the import and export filters. Second, the policy should be consistent when checked against Internet routing, with a minimum requirement to register import and export rules for its neighbors in the path. We consider an AS as having a **good policy** if we can match a certain percentage of its policy, and we can check a percentage of the paths for policy correctness. If an AS doesn't have good policy we exclude it from our study. For simplicity, we use the same percentage as a cutoff point for both the policy and the percentage of paths we can check. We require a percentage of at least 80% of accuracy in order to classify an AS as having a good AS policy. The reason that we have a relaxed criterion is that it is common for large ISPs not to register the neighbors that have peer-to-peer business relation. Since, we want to process as many ASes as possible we don't want to exclude them.

In table IV, we present some initial results for the three collectors. In rrc03, we found a total of $19,022$ ASes and $963,935$ unique paths. Out of these $19,022$ ASes, $10,313$ register their policy and thus we can check the validity of the paths that contain them. In total for all ASes with policy we have $3,754,159$ path tuples to check. This means that we can check each path using more than one AS. We find that 44.8% of these path tuples can be validated. When we restrict the ASes and use only those with good policy, we have $8,322$ ASes and we can check $1,597,545$ path tuples. We can validate 79% of these tuples. For rrc03 and ASes with good policy, we have $131,394$ path tuples that can be fully validated which is 10% of the total path tuples validated. The number of ASes with no flags are 4415 for rrc03, 4345 for rv2 and 5075 for rrc06.

Most ASes with no path flags are medium to small sized AS. In Figure 15, we plot the degree of the ASes that have good policy, and the degree of the ASes that have good policy and have no path flags. Most of the ASes with no flags are medium sized ASes, or customer ASes. Almost all larger ASes with good policy have both flags caused by not registering links and by incorrect policy. We investigated the policy flags and found that most of the policy flags are caused by as-sets in the aut-num policy that are not kept up to date. The sets used in the policy are hierarchical and thus while an AS might update its own set, the set might contain other sets in a deeper level that are not updated.

Next, we study the ASes per RIR. We classify the ASes based on the RIR that allocated them. For example if an AS is allocated in ARIN, but the policy of the AS is in RIPE, we would include it in the ARIN region. In Figure 16, we have per RIR, the number of ASes that are found in BGP, register policy, have a good policy, and have no flags. Again, we find that the best RIR is RIPE. In RIPE, we have that 93% of the ASes have good policy, while for 60% of the ASes we can validate all the path tuples that go through them. For APNIC the percentages are 69% and 40%, for ARIN 7% and 4%, and for LACNIC 6% and 4%. Basically, these percentages show that most of the ASes in ARIN and LACNIC region don't register any policy.

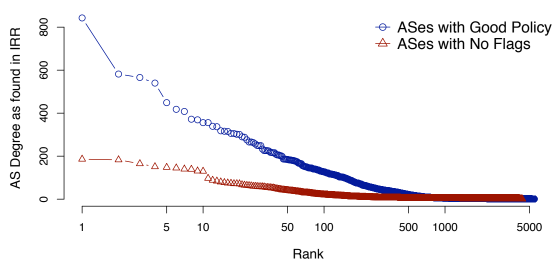

Fig. 15. The degree of ASes (IRR) that have a Good policy.

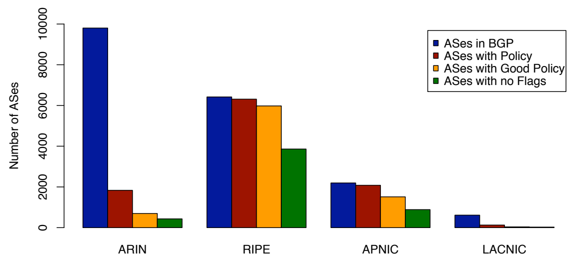

Fig. 16. ASes for various RIR

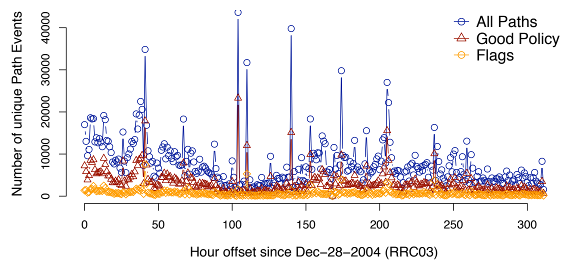

Fig. 17. Evolution of Events for the Path Policy.

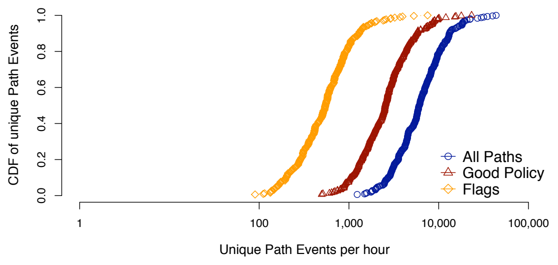

Fig. 18. Evolution of Events for the Path Policy(rrc03).

D. *Reactive path validation*

Next, we evaluate how useful our approach can be for validating the AS path. We use the same methodology as we did in the origin AS validation. We start with the routing table of December 24 2004, and we assume that we have validated all the paths that exist in the routing table. Then, we start processing the updates. We attempt to validate every new path, we call this an **event**. We validate the path for all the ASes in the path with good policy.

Caching alone won't help the path validation. In Figure 17, we plot the evolution of events versus time. First, note that we consistently have over a few thousands of unique paths per hour, with spikes reaching over $40,000$ new paths. Additionally, the numbers of new paths does not seem to decrease with time, and thus we can make the same conjecture with the origin AS flags, that using a cache to validate the paths is not an effective approach. Using the policy in IRR helps lowering the number of events an administrator needs to manually validate, but still there are too many flags.

Path validation requires more accurate policy. In Figure 18 we plot the CDF of the path events per hour. We find that for the ASes with good policy we always have more than 100 flags per hour. This suggests that the highly dynamic nature of paths and the fact that most large ISPs don't update their policy frequently makes it unrealistic to check the paths in a wide scale. Path validation, under the current registered policies, can work only for the ASes that maintain a 'perfect' policy.

To summarize the path validation results, we find that a reactive approach is possible but only for a limited number of ASes. There are at least 4000 such ASes, but most of them are small ASes that have a limited number of peers. There is a significant number of larger ASes that have reasonable policy, but the information they register is not sufficient to monitor their operation with reasonable overhead.

V. The profile of a Major Routing leak

In this section, we study an actual leak that occurred in December 24 2004. We use the routing collector rv2 to study the leak. At 9:29 UTC time, an AS from Turkey, AS9121 by mistake advertised to its neighbors over $100,000$ prefixes. This was the largest single incident since the AS7007 leak in 1997. The AS9121 leak gives us a unique opportunity to examine the reaction of ISPs and observe the behavior of the system. It is similar to studying the frequency response of a system, which is typically measured by applying an impulse to the system and measuring its response. This leak was so large that every ISP should have identified it within few minutes.

What happened? The source of the leak AS9121 advertised to its peers over $100,000$ prefixes. Usually, ASes accept from a peer a maximum number of prefixes to limit the damage for exactly these kind of incidents. This was the case for example with AS1239 and AS1299. Unfortunately, this was not the case with AS6762, which accepted everything AS9121 advertised to it. In total we found that 90% of all paths in rv2 were propagated by AS6762, while only 7.3% by AS1239 and 2% by AS1299. Note, that we calculated this percentage by using the AS that is adjacent to AS9121 in the AS path. The reason we mention this is that for example AS1239 also propagated bad prefixes it learned from AS6762.

There was virtually no warning for the leak: In figure 19, we plot the number of origin AS flags for AS9121 versus time. We start our evaluation in December 20, 2004, 4 days before the leak. As shown in the figure, there is only a single spike at the hour that the leak happened[9]. AS9121 created no flags before the main incident. This means that there was no warning that something was going to happen, and thus the ISPs were unprepared.

Duration of Incidents: Interestingly, AS9121 created two

[9]Note, that a flag is counted only once the first time it appeared. As we will talk later AS9121 created two incidents.

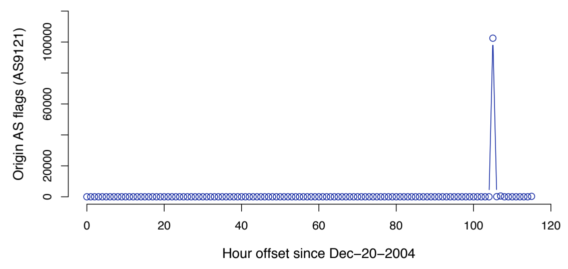

Fig. 19. Evolution of flags that AS9121 originates.

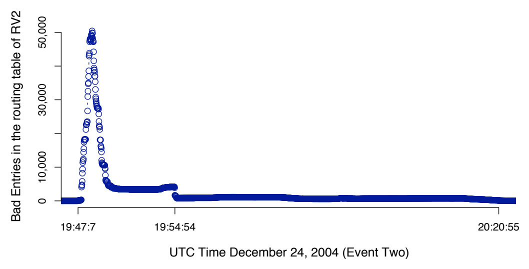

Fig. 21. Bad entries in the routing table or rv2 for the second round.

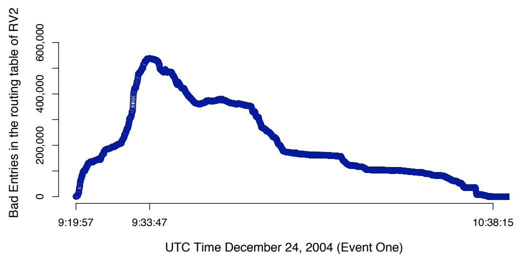

Fig. 20. Bad entries in the routing table or rv2 for the first round.

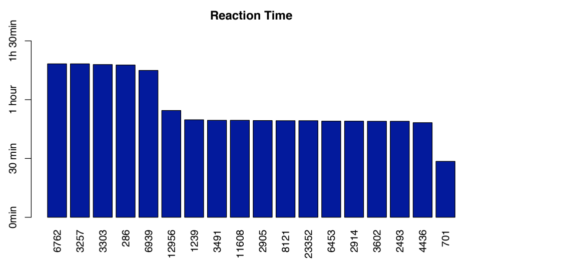

Fig. 22. Reaction time for ASes that appear in at least $10,000$ invalid entries.

rounds of incorrect advertisments. In figure 20, we have the first round. We plot the evolution of the number of bad entries in the routing table of rv2. The round started at 9:19:57 and peaked at 9:33:47 with close to $600,000$ bad entries. This figure shows that for the first round we had a duration of over one hour. In figure 21, we plot the second round that started at 19:47:7, and peaked at $50,000$ bad entries. The second round was much smaller than the first. This shows that when ASes anticipate possible leaks the leak incident becomes much smaller and shorter.

ASes reacted slowly: In figure 22 we plot the reaction time for a total of 18 ASes. The figures shows that most of the ASes reacted very slow. Their reaction time has over an hour. The AS that reacted first was AS701, but it took half an hour. Basically, the bad entries were withdrawn from the routing table when AS6762 stopped advertising the bad prefixes.

VI. Discussion: What can we do next

In this section, we first discuss how ISPs can improve their reaction to routing problems. Then, we discuss what should be improved in the registries, both RIR and IRR, in order to improve their usability. In the end, we discuss the importance of a reaction scheme.

A. A Systematic reaction approach for ISPs

In the previous section, we saw indications that some ISPs may not handle problems effectively, and they react too slowly. We believe that the ISPs should be prepared for these kind of situations, since these are not uncommon. Our goal is to limit the impact of routing errors and go from problems that last hours to problems that last seconds or in the worst case minutes.

Reaction to origin AS flags: In order to react, an ISP will have to answer the following questions: a) Is it a big or a small event? b) Does it involve my own prefixes or prefixes of my customers? c) Will I use a conservative approach? In table V, we have a high-level initial decision table for an ISP that can be elaborated and fine-tuned further. Based on the conditions, we have a number of rules that determine the actions and the sequence of actions for the ISP. For example, rule $r1$ is invoked if (a) we have a small event that (b) includes prefixes that the ISP originates. The action that correspond to rule $r1$, is first to deaggregate the prefixes that are affected, and then apply filters based on prefixes to block the leak[10]. This way even if it takes hours to block the original leak, the more specific prefixes that the ISP advertised will guarantee that no traffic will be lost. Of course, after the end of the event the ISP should withdraw these prefixes. A problem can arise with the length of the deaggregated prefixes. For example, if we own a $/19$, and someone leaks our prefix, we can advertise more specific prefixes than the leak, thus two $/20$, and we can solve the problem. This may not work if the hijacked prefix is a $/24$. Advertising two $/25$ will not necessary solve the problem since most ASes will not accept such specific prefixes. The only possible way to solve this problem is to use some sort of special BGP community with universal meaning.

There are several other issues that our systematic reaction needs to address. First, how we define an event as small. We use the size of the event to differentiate how we will block the leak, either by using filters on prefixes or using filters on path. For this first case, we can leave this for the ISPs to decide, since it depends on many other things such as their infrastructure. We also use the size of the event to decide if we want to deaggregate our prefixes. In the case of 9121, if everybody decided to deaggregate during the event, the routing tables would be flooded with more specific prefixes and thus it could probably cause worse problems in the Internet. Thus, the size of the event should be small enough to guarantee that there would be no danger for the Internet at large. Given that

[10]Note, that we apply filters only in the case of a local update.

TABLE V
DECISION TABLE FOR AN ISP ON HOW TO REACT TO ROUTING LEAKS

	Rules			
Conditions	r1	r2	r3	r4
Small Event	Y	Y	Y	N
Own Prefixes	Y	N	N	-
Conservative	-	N	Y	-
Actions				
Filter based on Prefixes	2	1	2	-
Filter based on Path	-	-	-	1
Deaggregate	1	-	-	-
Confirm Leak	-	2	1	-

the routing tables currently have over $160,000$ prefixes, a few thousands of additional prefixes for a single event should be fine.

Another implementation issue to improve our scheme is the selection of peers to apply the filters, other than the peer we received the bad announcement. We can either apply the filters on demand, or using the existing routing table we can find from which peers we receive legitimate announcements from the AS that generated the flags. This way, we can anticipate from which peers we might receive bad announcements and proactively apply the filters.

The last issue is how an ISP should decide whether to be conservative in its reaction. If we are conservative, first we need to investigate and then block the update if it is flagged. On the other hand, if we are not conservative, we can automatically block the flags and then investigate.We believe that a non conservative mode is more appropriate since we showed that the number of flags per hour is extremely low. The key for this approach to work, is a global whiteboard an ISP can use to notify other ISPs that it blocked the announcement. We believe that it should be the responsibility of the origin AS to update their records to prove that they can be the origin, and not the responsibility of the AS that blocked the announcement to investigate it. This could also motivate ISPs to keep their registry records up-to-date.

Reaction to path flags: For the path flags, we assume that an AS identified a path that contains its AS number and violates its registered policy. For example, a customer AS may act as transit to another provider. Unfortunately, the reaction to the policy flags is severely limited. For example, deaggregating won't help since BGP is a dynamic protocol and the AS that causes the problem will act as a transit for them also. The only possible solution is to try to contact the AS that causes the problem in order to fix it. An extreme measure might be to stop advertising the prefixes that appear in the bad paths, if the AS that causes the problem is a direct neighbor.

B. How can we improve the registries?

In the previous sections, we identify some subtle issues with the registries that are the cause of inaccuracies and inability to use the information effectively. These issues were not evident prior to our analysis, and we argue that these are the first issues that can and should be fixed.

RIR specific improvements: First, ARIN could prevent the unnecessary use of organization records in its registry. There exist close to one million organizations in ARIN. Practically, for every AS number or IP prefix, an ISP creates a new organization. The side effect is that we can not always find the correlations between the AS numbers and IP prefixes. The fix here can be that only ARIN can create new organizations, so that the hierarchical nature of the registration is maintained.

In addition, it would be important to disambiguate the organizations at a global scale. There exist a lot of organizations that operate across many RIR regions, the ideal case will be to have a unique ID across regions and administrative domains.

Second, RIR could improve the exchange and interoperability of information with the national Internet registries(NIR) that operate within their region. APNIC and LACNIC are the only RIR that allow the operation of NIR. In the APNIC area there exist four NIR that allocate resources within their country limits. The problem with these registries is that they are not transparent, and some of them don't register the direct top allocations to the ISPs. The ISPs do register their assignments, but this is not sufficient to analyze the allocations. Additionally, most of these registries have no records about organizations and AS number allocations. Another problem with not registering the top allocations is that we can't find the prefixes that are not allocated yet. As a result, these prefixes can easily be hijacked. In LACNIC, we have similar problems. For example, the brazilian NIR doesn't provide the necessary bulk whois data, that is, their allocation records.

Third, RIR can contain conflicting information registered by their administrators. This is surprising considering that RIR demand that the ISPs maintain accurate information. An example of a conflict is the registration records for AS513. This is the AS number for CERN, the European Organization for Nuclear Research. The RIR that should have the authority on this AS is RIPE, but in the RIPE registry the records indicate that CERN is under the authority of ARIN. On the other hand ARIN correctly points to RIPE.

General improvements: RIR and IRR have different objectives, but often they coexist in the same database as is the case with RIPE and APNIC. Originally, these registries were separate and there was a design decision to unite them for a more complete and better policy description. Using the RPSL format, one can describe the policy of an AS and additionally correlate its policy with the allocation records. This is crucial for validating the routing, thus the motive for this move was well justified. Additionally, we show that RIPE and APNIC perform better than ARIN. In reality though it never worked well. The percentage of origin AS tuples that can be strongly validated is only 73% for the case of RIPE. We believe that mixing allocations records and policy records, that usually contain inaccurate or stale information reinforces the belief that the information in the registries is incorrect. This can be dangerous for a system that is based on the quality of information. If we allow small problems, these will escalate quickly to major problems as the theory of broken windows proclaims [11]. We believe that either the RIR should enforce more strict rules on the policy records, or under the current conditions the registries should be separated. In the latter case, the RIR registries should be augmented with records similar to the route records in IRR. In addition, we believe that a new record or attribute should be introduced to authorize the use of as-sets and AS numbers in an as-set. Sets are used to describe what an AS will transit, and if no restrictions are applied an AS can claim that is the transit of any AS.

C. Why is a reaction scheme necessary?

Here, we present two problems that even though they are small today, they can create significant problems for the Internet in the future.

Permanent deaggregation because of hijacking: It was recently discussed in the NANOG mailing list that Covad, AS18566, has deaggregated their prefixes. Under normal operation they could originate 6-9 prefixes, but they originate 817. The explanation was that some AS in eastern Europe deaggragated and advertised with no permission some of their prefixes and thus they lost a significant amount of traffic. Their solution was not to allow this to happen again by permanently deaggregating their prefixes. Our solution provides a way to react to the problem by deaggregating on demand and not permanently. This way we can maximize our goal without polluting Internet routing tables.

Unauthorized use of resources: There are a number of prefixes and AS numbers that even though appear to be unallocated appear in routing tables. We can find 83 prefixes in BGP that are administered by ARIN that are not allocated. For RIPE we can find 11 such cases. For example, we can find in the routing tables prefixes like 64.57.160.0/19 which falls within ARIN. In the ARIN registry, we can find 64.57.144.0/20 and 64.57.192.0/19, which contain the immediate previous and following allocation records. Similarly for the AS numbers, in rrc03 we can find 58 ASes that don't have any registration records, for example like $AS1478$ and $AS29302$. Under a reactive scheme these cases will be identified immediately and thus blocked.

VII. Conclusions

Our work suggests the use of reactive approaches until (if ever) an ultimate solution for BGP robustness appears in the future. As our main contribution, we develop an approach and a tool for validating the origin and path of a BGP update. The method is ready-to-use: it can be deployed today, with the currently available information. Our approach is intended to act as an advisor to a network administrator. In addition, we apply our method to real data, and we propose ways to improve the capability of ASes to respond to routing errors.

Our approach is an effort to develop tools of immediate practical applicability. The approach could act as an aid to network administrators, identifying suspicious updates which is only a small fraction of the total updates. In fact, even if we just monitor the number of flagged updates, an sudden spike could signify that something is wrong. By applying our tool on real data, we arrive at three high-level observations.

A. Registries are useful. The registries contain enough information to be very useful even as they are, although careful processing is needed.

B. Small effort, big pay off. Small modifications and attention at improving the information of the registries can have significant impact in our ability to safeguard BGP routing.

C. ASes are unprepared. Many ASes do not seem well prepared to handle routing misbehavior. We saw that the reaction time for a large scale event (which should have been easier to detect) took hours. We conjecture that smaller scale events, which can be more frequent, may take longer to detect, if they ever get detected.

In the future, we expect to further develop our approach into a comprehensive automated tool, which could act as the first line of defense to routing misbehavior.

References

[1] Internet Routing Registries. http://www.irr.net/.

[2] Routing information service(ris). www.ris.ripe.net.

[3] Routing protocols security working group. http://www.rpsec.org/.

[4] Anatomy of a leak: AS9121 (or, 'how we learned to start worrying and hate maximum prefix limits'). *NANOG 34*, 2005.

[5] Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T.Bates, D. Karrenberg, and M.Terpstra. Routing Policy Specification Language(RPSL). RFC2622.

[6] R. Chandra, P. Traina, and T. Li. BGP Communities Attribute. RFC1997.

[7] S. Convery, D. Cook, and M. Franz. An attack tree for the border gateway protocol. Internet Draft.

[8] D.Meyer, J.Schmitz, C.Orange, M. Prior, and C. Alaettinoglu. Using RPSL in practice. RFC2650.

[9] N. Feamster, J. Winick, and J. Rexford. A model of BGP routing for network engineering. *IEEE Sigmetrics*, 2004.

[10] V. Gill, J. Heasley, and D. Meyer. The generalized TTL security mechanism (GTSM). RFC3682.

[11] C.M.Coles G.L. Kelling. *Fixing Broken Windows*. The Free Press, 1996.

[12] Geoffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick McDaniel, and Aviel Rubin. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. *Symposium on Network and Distributed Systems Security*, 2003.

[13] A. Heffernan. Protection of BGP sessions via the TCP MD5 signature option. RFC2385.

[14] Yih-Chun Hu and Adrian Perrig. SPV: A Secure Path Vector Routing Scheme for Securing BGP. *ACM Sigcomm*, 2004.

[15] Ng James. Extensions to BGP to support secure origin BGP (sobgp). Internet Draft, 2002.

[16] Stephen Kent. Securing the border gateway protocol: A status update. *Seventh IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, 2000.

[17] Stephen Kent, Charles Lynn, and Karen Seo. Secure border gateway protocol (S-BGP) architecture. *IEEE JSAC Issue on Network Security*, 2000.

[18] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfigurations. *ACM Sigcomm*, 2002.

[19] Stephen Misel. Wow, as7007! http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html.

[20] Sandra Murphy. BGP security vulnerabilities analysis. INTERNET DRAFT, 2003.

[21] Sandra Murphy. Routing protocol threat analysis. INTERNET DRAFT, 2003.

[22] W. B. Norton. The art of peering: The peering playbook. *Draft*.

[23] University of Oregon Route Views Project. Online data and reports. http://www.routeviews.org/.

[24] Dan Pei, Lixia Zhang, and Dan Massey. A framework for resilient internet routing protocols. *IEEE Network*, 2004.

[25] Shared Whois Project. http://www.arin.net/reference/database.html.

[26] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771, 1995.

[27] Georgos Siganos and Michalis Faloutsos. Analyzing BGP policies:methodology and tool. *IEEE Infocom*, 2004.

[28] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy H. Katz. Listen and whisper: Security mechanisms for BGP. *First Symposium on Networked Systems Design and Implementation*, 2004.

[29] C. Villamizar, R. Chandra, and R. Govindan. BGP route flap damping. RFC2439.

[30] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wua, and Lixia Zhang. Detection of invalid routing announcement in the internet. *International Conference on Dependable Systems and Networks (DSN'02)*, 2002.