Computer Networks 211 (2022) 108966

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Practicable route leak detection and protection with ASIRIA

Marcelo Bagnulo[a], Alberto García-Martínez[a,*], Stefano Angieri[a], Andra Lutu[b], Jinze Yang[c]

[a] Dep. Ing. Telemática, UC3M, Spain
[b] Telefonica Research, Spain
[c] Huawei Technologies Co., Ltd., China

ARTICLE INFO

Keywords:
BGP security
Route leaks
Routing registries

ABSTRACT

Route leak events have historically caused many wide-scale disruptions on the Internet. Leaks are particularly hard to detect because they most frequently involve routes with legitimate origin announced through legitimate paths that are propagated beyond their legitimate scope. In this paper we present ASIRIA, a mechanism for detecting and avoiding leaked routes and protecting against leakage events that uses AS relationship information inferred from the Internet Routing Registries. By relying on existing information, ASIRIA provides immediate benefits to early adopters. In particular, we consider the deployment of ASIRIA to detect leaks caused by over 300 ASes and we show that it can detect over 99% of the leakage events generated by a customer or a peer solely using currently available information in 90% of the cases.

1. Introduction

Over the last decade, a vast amount of effort has been devoted to secure the Border Gateway Protocol (BGP), leading to the definition of several BGP security mechanisms. Notably, the Resource Public Key Infrastructure (RPKI) [1] provides the means to perform *origin validation*, i.e., to validate the AS originating a route, and BGPSEC [2] enables *path validation*, so that the sequence of ASes through which a BGP route is propagated over the Internet can be verified. While these two mechanisms protect against several types of common attacks such as prefix hijacks, they fail to protect from another set of very common threats known as *route leaks*.

In June 2015, Telekom Malaysia (AS4788) leaked (i.e., announced) over 179,000 routes learned from providers and peers to one provider, Level3. The provider, preferring routes received from customers over routes received from peers, switched its forwarding path for these destinations to AS4788, and in turn, propagated the new routes to its own peers and clients. The affected prefixes included Google, Microsoft, LinkedIn and Reddit. Preferring leaked routes resulted in severe performance degradation when attempting to communicate with these destinations [3]. This is just one of the numerous route leakage incidents that occurred over the last 20 years [4], many of which resulted in major disruptions of the Internet service affecting large geographical areas.

A *route leak* is defined as the propagation of a route beyond its intended scope [5]. For example, as depicted in Fig. 1, AS1 is multihomed to two providers, ISP1 and ISP2. If AS1 announces the routes learned

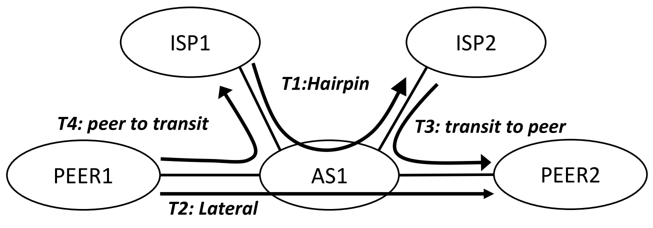

Fig. 1. Route leak scenarios.

from ISP1 into ISP2, this would result in a route leak, since the routes announced by a provider are (in general) intended to be distributed to the customer and to the customer's clients but not to its providers. This simple example also illustrates why RPKI and BGPSEC combined fall short in protecting from route leaks. Indeed, in the aforementioned example, the routes announced by AS1 to ISP2 are generated by the legitimate holder of the prefix (so RPKI/origin verification is successful) and each router along the path has propagated them through valid relationships between ASes (so the BGPSEC/path validation also succeeds). In general, route leaks are the result of propagating routing information with a valid origin and acquired via a valid relationship beyond the scope intended by the routing policies of the involved ASes.

The Internet Routing Registries (IRRs) are repositories where ASes declare their routing policies and as such, they can be used to support

* Corresponding author.
E-mail address: alberto@it.uc3m.es (A. García-Martínez).

https://doi.org/10.1016/j.comnet.2022.108966
Received 18 October 2021; Received in revised form 24 February 2022; Accepted 11 April 2022
Available online 18 April 2022

route leak detection and protection. According to the current Mutually Agreed Norms for Routing Security (MANRS) guidelines [6], operators are encouraged to use the information available in the IRR to create filters that protect against route leaks. Specifically, the MANRS guidelines on filtering recommend operators to require their direct and indirect customers to register their prefixes and AS numbers in an IRR and use this information to create filters that discard any route announced by their customers that does not match with the registered prefixes and AS numbers. Thus, a network uses the information inserted in the IRR by its customers to generate an *accept-list* that includes all valid prefixes and AS-paths. The implication is that all routes that are not explicitly declared in the *accept-list* (even if they are legitimate) are filtered out. The Internet community has been pushing for the universal deployment and configuration of such filters, but the large number of route leak events that we continue to witness on almost a daily basis show that the correct configuration of such filters is far from universal. This is likely to be due to a variety of reasons. While this approach may be practical for ISPs closer to the edge, it becomes much more challenging for ISPs in the core with multi-tier customer cones. In this case, the complexity of the relationships and the dynamic nature of the routing information makes it difficult to keep fresh information suitable for the configuration of filters affecting all the ASes in the customer cone. Moreover, filters configured by ISPs in the core are less effective than those configured by ISPs in the edge, because they do not protect against route leaks within the customer cone, which in this case is fairly large. Finally, filtering out legitimate customer routes because they have not been properly declared in the IRR may impact customer satisfaction and may have negative commercial consequences.

In this paper, we propose an alternative paradigm on the use of IRR information to protect from leaks, from the current *accept-list* one to another one based on *drop-lists*. We introduce Autonomous System Internet Registry Inference for path Authorisation (ASIRIA), a novel mechanism to provide protection against route leaks based on *drop-lists*. ASIRIA infers the type of relationships between ASes from the information available in the IRR. Using the information about ASes' relationships, ASIRIA identifies and avoids leaks. Leaked routes are detected by analysing the relationships between consecutive ASes present in the AS_PATH attribute of a BGP route. Valid routes contain *valley-free* paths[1] [7] while leaks have AS paths that are not *valley-free* [5]. ASIRIA identifies routes containing non *valley-free* paths and includes them in the *drop-list*, in order to avoid using them if alternative routes are available. ASIRIA also keeps track of the number of leaks detected and uses this information to build a leakage event alarm mechanism that allows ASIRIA-enabled routers to detect leakage events and notify the network administrator of suspicious behaviour.

The effectiveness of the *drop-list* approach depends on the quantity and the quality of the information available in the IRR and mobilising a large fraction of the ASes in the Internet to populate the IRR is indeed challenging. Thanks to the long-lasting efforts of the Internet operational community in promoting the population of the IRR, a significant number of ASes have already registered routing policy information in the IRR. We show that using current IRR information we can infer 97k AS relationships with a precision of 96,7%. With this information already available in the IRR, we compute the ability of ASIRIA to detect leaked routes using real routing information from 302 ASes. We find that the ASes implementing ASIRIA can detect and reject an average of 17% of the leaked paths in case a customer or a peer leaks the full BGP routing table, and that the top 20% of the measured ASes can detect at least 40% of leaked paths. While more than half of the monitors detect less than 10% of the leaks, we show that leak detection can be greatly improved by properly annotating in the IRR the relationships for a small fraction of the AS pairs currently observed in the routes (we can improve leak detection more than 80% by annotating just 0.3% AS pair relationships). Regarding the detection of leakage events, the results are very positive, as 90% of the analysed ASes can detect more than 99% of the leakage events simulated in our analysis. These results show that ASes implementing ASIRIA can obtain immediate benefits by leveraging on existing IRR information and that this can drive ASIRIA adoption, fostering other ASes to populate the IRR, enabling a virtuous cycle of IRR population and ASIRIA adoption.

The rest of the paper is structured as follows: We first describe route leaks and its different types. Section 3 presents an overview of the ASIRIA mechanism, describing its components. In the next section we describe a method to infer AS pair relationships from IRR information and we evaluate its performance comparing the results with those from the CAIDA dataset. Then, Section 5 is devoted to analyse the performance of ASIRIA using the inferred relationships and real BGP routes from different monitors. We estimate an upper bound for the number false positives detected and evaluate the ability of ASIRIA to detect leaks in the case of large leakage events. Then we consider how performance may evolve as more AS relationships (in addition to existing ones) were annotated. We also discuss the performance of ASIRIA's alarm system. In Section 6 we discuss ASPA, an alternative method to ASIRIA being currently developed by the IETF, and how ASIRIA and ASPA can be combined to detect leaks. Finally, we discuss related work in Section 7 and present the conclusions in Section 8.

2. Route leaks

The most common relationships established between two ASes are *provider to customer* (p2c) and *peer to peer* (p2p) [8]. In the former, the provider agrees to propagate all its routes to the customer, while the customer injects its own routes and those from its customers into the provider. The intention of this settlement is to allow the customer to access the Internet through its provider. In a p2p relationship both participants exchange their own routes and those from their customers, enabling shortcuts between the involved ASes. In this way, they avoid transiting through their respective providers. There are other relationships, such as siblings or partial transit, but they are much less common than the ones described above [9].

RFC 7908 [5] defines 4 types of route leaks according to the order in which p2p and/or p2c relationships appear in the AS path attribute of the route announced through BGP. As illustrated in Fig. 1, the four types of route leaks are the following:

- *Type 1 — Hairpin*: A (multihomed) client announces a route learned from one provider to another provider. The AS path of the leaked route contains a p2c relationship, followed by a c2p relationship.
- *Type 2 — Lateral*: An ISP announces a route learned from a peer to another peer. The AS path of the leaked route contains (at least) two p2p relationships.
- *Type 3 — Transit to peer*: An AS announces routes learned from a provider to a peer. The AS path of the leaked route contains a p2c relationship followed by a p2p relationship.
- *Type 4 — Peer to transit*: An AS announces routes learned from a peer to a provider. The AS path of the leaked route contains a p2p relationship followed by a c2p relationship.

Note that all leaks are generated when a route is advertised either to a provider or a peer, but never when a route is advertised to a customer.

Additionally, RFC 7908 defines route leaks of type 5 and 6, but these leaks are related to mis-origination or re-origination of routes by the offending AS, so they can be prevented by existing techniques such as

[1] Valley-free paths follow a pattern in which packets go from a customer to a provider (possibly many times), then may go to a peer, and descend to a customer (possibly may times) until the destination is reached. In this pattern, every AS in the path has an incentive for forwarding the traffic. Subsets of this pattern are also reasonable from the economic perspective. Non valley-free paths are those with any other pattern.

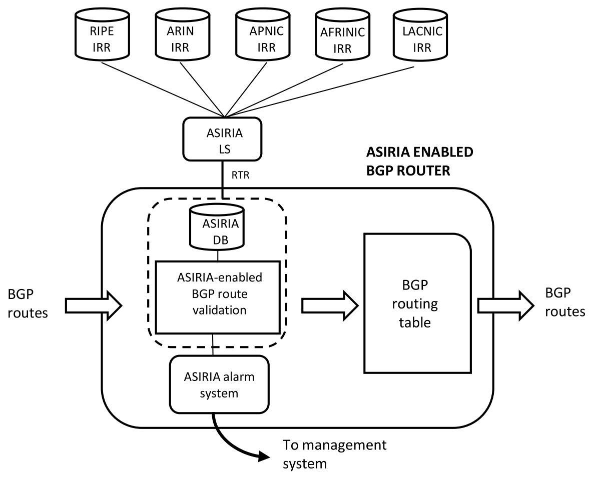

Fig. 2. ASIRIA architecture.

RPKI-based origin validation. In the rest of this paper we only consider route leaks of types 1 to 4.

While it is possible to leak a single route, it is not uncommon that route leak incidents involve a large number of prefixes (e.g., 179K routes were leaked in [3]). For the purpose of this paper, we use the term *route-leak* when referring to the announcement of each individual route, and we use the expression *(route) leakage event* when referring to an incident that involves multiple routes being leaked. It is possible to discover a route leakage event by detecting some of the route leaks involved without having to detect each and every one of the route leaks involved in it.

3. The ASIRIA mechanism for route-leak protection

ASIRIA is a mechanism for route leak detection and protection. ASIRIA has three main components: (i) the **ASIRIA Inference Algorithm** to infer relationships between ASes from the information available in the IRR, (ii) the **ASIRIA Route Validation**, that ASIRIA-enabled routers run to perform real-time detection of invalid routes using the inferred AS relationship information, and (iii) the **ASIRIA Alarm System** that detects leakage events.

The ASIRIA Inference Algorithm extracts c2p and p2p relationships from the IRR using the `import` and `export` family of attributes present in the `aut-num` objects. In Section 4.1, we detail the process to infer each type of relationship.

The *ASIRIA Local Server* (ASIRIA LS) executes off-line the ASIRIA Inference Algorithm that mines the IRR database to extract the relationships between neighbouring ASes, as we show in Fig. 2. The information the ASIRIA LS obtains is in the following form:

(AS1, c2p, AS2), indicating that AS1 is a customer of AS2.

(AS1, p2p, AS2), indicating that AS1 and AS2 have a peering relationship.

The ASIRIA LS conveys this information to an ASIRIA-enabled BGP router. The mechanism to transfer the information from the ASIRIA LS to the router is similar to the RPKI-RTR protocol [10] (currently used by routers to install the information originated in the RPKI). The router stores the ASIRIA information in the *ASIRIA Data Base* (ASIRIA DB). In addition, the network manager should feed information about the relationships with neighbouring ASes, if such relationships cannot be inferred from the IRR data.

Using the information available in the ASIRIA DB, an ASIRIA-enabled router executes the ASIRIA Route Validation process. When it receives a BGP route, it extracts the pairs of ASes included consecutively in the AS_PATH attribute of the route, and searches for matches in the ASIRIA DB. It then searches for patterns of AS relationships present in the AS path that would imply that the route is a leak, as discussed in Section 2. Such routes are deemed invalid. In many cases, the relationships for one or several of the AS pairs present in the AS path is unknown. These AS pairs are simply ignored, and the validation process continues to look for other indications that the route may be invalid.

Table 1
Route leak types.

Leak Type	AS relation sequence
1	...p2c...c2p...
2	...p2p...p2p...
3	...p2c...p2p...
4	...p2p...c2p...

Table 1 lists the patterns used by ASIRIA to detect each leak type.

If a route is marked as invalid, the router decreases its priority, so that the routing algorithm prefers other valid routes for the same destination. The reason for not plainly discarding invalid routes is that the information on the IRR may be wrong (e.g., outdated) or incomplete, causing the ASIRIA algorithm to mis-infer a relationship. By de-prioritising invalid routes, ASIRIA is able to select alternative valid routes (if they exist), but still use routes marked as invalid if no alternative route is available. So, by design, ASIRIA assumes that there may be errors in the inferred relationships, and it tries to limit the effects of such errors. This design choice allows ASIRIA to work with inaccurate information.

In addition to de-prioritise specific invalid routes, ASIRIA also detects leakage events and triggers an alarm that can serve as an early warning to the system administrator. The ASIRIA Alarm System works as follows: ASIRIA maintains statistics about the number of preferred routes that are marked as invalid. In normal operation, when there is no ongoing leak, this number is expected to be small, accounting for errors in the AS relationship inference mechanism. The ASIRIA leakage event is triggered when the number of preferred routes that are marked as a leak suddenly increases beyond a configured threshold, i.e., a 100% increase in a 5-minute period (as described below in Section 5.5).

4. Description and validation of AS pair relationship inference

Over the last 20 years, there has been a lot of effort devoted to the discovery of the Internet topology, in general, and to the inference of the type of relationships between ASes, in particular. Notably, the work done by CAIDA [11] achieves a very high accuracy when inferring relationships (99.6% in the case of c2p relationships and 98.7% for p2p relationships). It would then be natural to use this (or other similar approach) to feed the ASIRIA DB with information about the type of relationships between ASes. Unfortunately, these approaches are unsuitable for the purposes of ASIRIA. The reason is that the relationship inference algorithms such as the one proposed by CAIDA rely on topological characteristics collected from multiple sources. While the output of these algorithms is highly accurate, they still have a small number of mis-inferences. The problem is that in these cases, the affected ASes (whose relationship has been miss-inferred) have no knob available to correct the error. Consequently, they will suffer the effects of miss-configured filters without being able to do anything about it. It is then mandatory that the mechanism ASIRIA uses to infer relationship between ASes provides the means to the involved ASes to correct any relationship in which they are involved. For this reason, ASIRIA infers the AS relationships from the IRR. Any error in the IRR database can be corrected by the involved ASes. We do use the CAIDA dataset (which is highly accurate) to contrast and validate the relationships inferred by ASIRIA using the IRR information.

4.1. Criteria to infer relationships between ASes using IRR data

ASIRIA uses the data available in the IRR to infer relationships between ASes. To do so, it relies on the information included by the ASes in their respective `aut-num` objects, in particular in the `import` and `export` attributes. The `aut-num` class is defined in [12] and it is used by an AS to declare its routing policy. The `import` and `export` attributes specify the import and export policies respectively.

As described in [13], in case of a transit relationship in which AS1 is a provider of AS2, the common configuration for the `import` and `export` attributes are:

(t1) `aut-num: AS1; import: from AS2 accept AS2 + AS2_customer_set`
(t2) `aut-num: AS1; export: to AS2 announce ANY`
(t3) `aut-num: AS2; import: from AS1 accept ANY`
(t4) `aut-num: AS2; export: to AS1 announce AS2 + AS_customer_set`

Through this configuration, AS1 states that it exports its full routing table while it only accepts routes originated by its customers and its customers' customers. AS2 states that it accepts all the routes announced by AS1, while it only exports a subset of the routes present in its routing table, typically its own and those belonging to its customers. We also account for alternative forms to express the same routing policy using the `mp-import`, `mp-export`, `default` and `mp-default` attributes. To simplify the exposition, and without loss of generality, for the rest of the description of the policies, we solely describe the `import` and `export` expressions, which are the most popular.

In the case that two neighbouring ASes have a p2p relationship, this is represented in the IRR as follows:

(p1) `aut-num: AS1; import: from AS2 accept AS2 + AS2_customer_set`
(p2) `aut-num: AS1; export: to AS2 announce AS1 + AS1_customer_set`
(p3) `aut-num: AS2; import: from AS1 accept AS1 + AS1_customer_set`
(p4) `aut-num: AS2; export: to AS1 announce AS2 + AS2_customer_set`

Through this configuration, both ASes express that they accept and announce prefixes belonging to the peering ASes and to their respective customer's cones.

In the case of a sibling relationship, when two ASes provide mutual traffic, this is represented in the IRR by both ASes declaring both import ANY and export ANY rules. However, a sibling relationship has no effect when detecting a leak, i.e., its presence or absence has no impact in the leak detection algorithm. Because of this, ASIRIA does not specifically infer sibling relationships, even though it would be possible to do so from the IRR data.

There are other more sophisticated relationships, such as *partial transit*, in which the provider offers transit to a subset of Internet destinations, and *hybrid relationships*, in which two ASes have different types of relationships in different interconnection points [9]. It is possible to describe these relationships using the `aut-num` objects above, but because they are less common (they account for 4.5% of the relationships according to [9]), we focus on inferring the p2p, p2c and c2p relationships that account for the vast majority of relationships.

4.2. Inferring relationships out of the IRR data

We next extract information about AS relationships from the IRRs. There are 25 IRRs according to IRR.NET. It is documented that the quality and the amount of data in the different registries varies heavily [14], as some of them contain stalled information and/or are inconsistent with each other. In this paper, we use the IRRs maintained by the RIRs (i.e., RIPE, APNIC, ARIN, AFRINIC and LACNIC), as they provide strong authentication means, guaranteeing that only the legitimate owner of the AS can update the information associated to its `aut-num` object.

We retrieve the IRR databases on the 2021-03-01. We find that most of the information comes from the RIPE database. It contains 644,039 policy expressions (`import`, `export`, `mp-import`, `mp-export`, `default` or `mp-default`), while the rest of the IRRs account for just 39,139 expressions. We filter out also those `import` and `export` attributes that contain the `refine` and `except` keywords, since they are very few of them and they are hard to parse (because of this, we filtered 650 objects out of the 600k expressions reported above). We expand the `as-set` attributes present within the retrieved `aut-num` objects.

The IRR data is frequently incomplete, as sometimes only one of the ASes involved in a relationship declares it in the IRR, or some ASes declare either the `import` or the `export`, but not both. Thus, we expect that it will be frequent that not all the four expressions are available for many pairs of ASes. We search for the pairs of ASes that match a subset of the conditions described earlier characterising the c2p and the p2p relationships.

An additional difficulty that we encounter when inferring relationships from the IRR data is that several of the expressions contained in the `aut-num` objects describing both c2p and p2p relationships refer to the customers of the respective ASes, which we have not identified yet. To tackle this, we start by identifying candidate sets of c2p relationships and p2p relationships by using conditions that do not assume any knowledge of inter-AS relationships. We then refine the inferred relationships in a later stage. The conditions for creating the candidate set for c2p relationships are:

(tc1) `aut-num: AS1; import: from AS2 accept filter != ANY`
(tc2) `aut-num: AS1; export: to AS2 announce ANY`
(tc3) `aut-num: AS2; import: from AS1 accept ANY`
(tc4) `aut-num: AS2; export: to AS1 announce filter != ANY`

The conditions for creating the candidate set for p2p relationships are:

(pc1) `aut-num: AS1; import: from AS2 accept filter != ANY`
(pc2) `aut-num: AS1; export: to AS2 announce filter != ANY`
(pc3) `aut-num: AS2; import: from AS1 accept filter != ANY`
(pc4) `aut-num: AS2; export: to AS1 announce filter != ANY`

In all these expressions `filter != ANY` means that the expression exists, and that the filter registered is different than ANY. Using these expressions, we create a candidate set for both c2p and p2p relationships.

We validate the relationships in the candidate sets (both for p2p and for c2p relationships) using the information about AS relationships from the CAIDA AS relationship dataset[2] [11].

With respect to c2p relationships, we find that there are 27,051 AS pairs that fulfil the 4 conditions (*tc1*, *tc2*, *tc3* and *tc4*). Among those also present in the CAIDA dataset, 98.9% match the type of relationship in both datasets; we thus conclude that these are suitable candidates for c2p relationships. For the AS pairs that match only 3 of the conditions (because the fourth one is not present in the IRR) and are present in both datasets, the match with the CAIDA dataset exceeds 95%, making them suitable candidates for c2p relationships. Among the AS pairs

[2] In order to validate the algorithm presented in [11] for generating the CAIDA AS relationship dataset, the authors used 3 datasets, direct reports from network operators, IRR data and information from BGP communities. The use of IRR data for validation creates a potential circular dependency. However, we observe that out of a total of 50,504 relationships they used for validation, only 6,530 were obtained from the IRR, while the remaining 43,974 were obtained from other sources.

Table 2
Number of inferred c2p relations and validation using CAIDA information. For each set of conditions detailed in the first column, the second column contains the total number of relations inferred from the IRR. The third column contains those relationships that are present in the CAIDA dataset and coincide with the relationship assigned by CAIDA.

Conditions				IRR	Matching with CAIDA c2p
tc1	tc2	tc3	tc4		
✓	✓	✓	✓	25,196	16,202/99.2%
✓	✓	✓		170	93/97.9%
✓		✓	✓	1,264	910/97.6%
	✓	✓	✓	1,074	612/99.7%
✓	✓			23,556	6,481/96.7%
	✓	✓		26	10/100%
TOTAL				51,286	24,308

Table 3
Number of inferred p2p relations and validation using CAIDA information. For each set of conditions detailed in the first column, the second column contains the total number of relations inferred from the IRR, the third column contains the number of those that are present in the CAIDA dataset and match with the relationship resulting from CAIDA's inference process.

Conditions				IRR	Matching with CAIDA p2p
pc1	pc2	pc3	pc4		
✓	✓	✓	✓	4,003	548/97.7%
✓	✓		✓	1,460	297/95.8%
✓	✓			41,136	4,937/95.1%
TOTAL				46,599	5,782

that match only two conditions, we observe that the results are not consistent. For AS pairs fulfilling *tc1 tc2*, *tc1 tc3* and *tc2 tc3* that are also present in the CAIDA dataset, the level of matching is higher than 95%, so they are also suitable candidates for c2p relationships. For the other pairs of conditions, the matching with the CAIDA dataset decreases and it is lower than 90%. This is expected as, for instance, the conditions *tc1 tc4* are not specific for c2p relationships (note that p2p relationships use the exact same expressions). We exclude these from the p2c candidate set. Consequently, for our c2p relationships dataset, we only select the relationships we inferred using the four *tc* conditions, all the combinations of three of the *tc* conditions and *tc1 tc2*, *tc1 tc3* and *tc2 tc3* as candidate set for c2p relationships.

With respect to the p2p relationships, we find that the relationships inferred using the four *pc* conditions, as well as the ones inferred using *pc1 pc2 pc4* and *pc1 pc2* have a matching with the CAIDA dataset higher than 95%. We thus select all of these as the candidate set for the p2p relationships. Other combinations have a matching lower than 90%, so we exclude them from the candidate set. We looked deeper into the set of relationships that matched the set of conditions *pc1 pc2 pc3*, since excluding them seems to be at odds with including the AS pairs that comply with *pc1 pc2*. We find that there is a very small set of AS pairs that comply with *pc1 pc2 pc3* (only 243 AS pairs). Out of these, only 45 of them are also present in CAIDA's dataset, 40 of them matching. We inspected closely those not matching with CAIDA's dataset, and we found that the mis-inference of relationships was related to a single (large) AS whose policies referred to an undefined AS-set. So, it is likely that AS pairs complying with *pc1 pc2 pc3* have indeed a p2p relationship, but given the small number of relationships inferred, a single error in a large AS causes a mismatch that results in excluding them from the analysis. In any case, we exclude them, as the impact in overall performance is negligible.

Once we have the initial candidate sets for both p2c and p2p relationships, we use the information provided by these candidate sets to remove the relationships that refer to a peer or to a customer in conditions *tc1*, *tc4*, *pc1*, *pc2*, *pc3* and *pc4*. After this process, we contrast against CAIDA information and verify the resulting set of p2c and p2p relationships. In Tables 2 and 3 we include the final number of p2c and p2p relationships inferred respectively as well as their matching with the CAIDA dataset. We infer a total number of 51,286 customer-provider relationships and 46,599 peer-to-peer relationships.

5. Performance evaluation of the ASIRIA mechanism

In this section we do a quantitative analysis of the performance of the ASIRIA solution. For this purpose, we use two metrics, the *sensitivity*, and the *false positive rate*. The *sensitivity* is the ratio between the leaks detected and the total number of leaks, and indicates how many of the actual leaked paths can be detected. The *false positive ratio*, which represents the number of valid routes identified as leaks divided by the total number of valid routes, reflects the probability of a *false positive*, i.e., how easy is for a valid path to be mistaken for a leak. We determine an upper bound for this value. In addition, we compute the sensitivity of the ASIRIA alarm system for detecting leakage events, i.e., events that generate a set of leaks.

5.1. Dataset

To compute the metrics in realistic scenarios we use real BGP routing tables from the RIPE Routing Information System (RIS). We retrieved the BGP routing tables from RIS on the 2021-03-01, 08:00. At that time, there were 1,601 monitors that periodically dumped BGP information to 20 collectors. The monitors have configured different routing/announcement policies, with some of them announcing their full BGP routing table to the collector. We select the monitors that provide a full IPv4 BGP feed to the collector. Following [15], we identify *full feeds* as those tables containing at least the 75% of the maximum number of prefixes observed in any RIS BGP feed. There are 302 monitors that comply with the 75% rule, with a mean number of advertised prefixes of 824,000 prefixes and 105,000 different AS path routes. We use these monitors in our analysis.

The ASIRIA DB contains 51K p2c relations inferred from the IRR, of which 24K also appear in the RIS BGP routing tables analysed. Regarding p2p relationships, there are 46K in ASIRIA DB, with 4.8K appearing in the RIS tables. The total number of different AS pairs appearing in the RIS BGP tables is 373K, so the fraction of these relationships also in the ASIRIA DB is 7.9%.

5.2. False positives in ASIRIA leak detection

We analyse the occurrence of false positives in ASIRIA (i.e., the leaks reported by ASIRIA that are not real leaks).

We run the ASIRIA leak detection algorithm in the BGP tables of the monitors in 2021-03-01. According to BGPstream,[3] there were no reports for on-going leaks at the time the BGP tables are retrieved (08:00). The mean number of invalid routes detected is 0.08%. The distribution of the invalid routes over the 302 IPv4 monitors considered is depicted in Fig. 3.

The mean number of routes marked by ASIRIA as a leak per monitor is 96.8 routes. We can compare this value with the mean number of routes that have two or more relationships identified per monitor, i.e., the routes for which it is possible to detect an invalid route, 8,491 routes. The ratio of false positives for this route set is 1.1%, and could serve as an estimation of the false positive ratio ASIRIA can achieve. This is consistent with previous work [16] that has found that 1.5% of the IPv4 routes are valley routes, some of which would be detected as leaks by ASIRIA.

We next inspect the routes identified as invalid, and we study their variation over time. We search for the same routes in the routing tables of the same RIS monitors, 8 h before and after. Most of the paths identified as invalid are stable and appear in the previous and subsequent routing table snapshots. More precisely, 95% of the routes

[3] https://bgpstream.com.

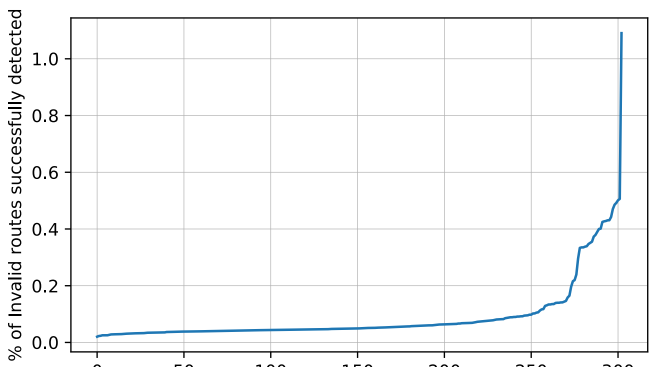

Fig. 3. Percentage of Invalid routes observed in RIS monitors over total routes.

marked as invalid observed at 08:00 were also present at 00:00, and the coincidence between 08:00 and 16:00 is 98%. We also consider the match with the routing tables captured 10 days after (2021-03-11, 08:00), to observe in this case a match of 85%. This suggests that most of the invalid routes of this set have not changed over time, probably indicating that they are legitimate routes (or at least, leaks that did not trigger any corrective action over time). This means that the false positive rate should be below the 1.1% upper-bound computed earlier.

ASIRIA is designed to take into account a non-negligible positive rate as the one we obtain. As routes that are leaks are not discarded, but they are de-prioritised, they are still used when no better alternative is available. This ensures that customer prefixes included in the aforementioned 1.1% are always reachable either directly or through other providers. Thus, ASIRIA does not discard legitimate traffic when there are false positives.

5.3. ASIRIA leak detection sensitivity

We compute the sensitivity of the proposed solution (the ratio between the leaks detected and the total number of leaks) when an AS leaks its full routing table to a provider or to a peer. The different leak scenarios considered are depicted in Fig. 4. In all cases, AS1 is running ASIRIA in order to detect the leaks. The scenarios considered include the leakage from a direct customer (AS C2 in the picture), any AS in the customer cone (AS C11 in the picture), a peer (AS P1 in the picture) and any AS in the peer's customer cone (AS CP1 in the picture).

For simplicity of the presentation, we first show the case of a leak from a direct customer as illustrated in Fig. 5. We generalise for other scenarios later on. In the considered scenario (Fig. 5), a router (L) in a customer AS (the *leaky neighbour*) leaks its whole routing table to an ASIRIA-enabled router (A) in its provider AS. We compute the level of protection provided by ASIRIA for this scenario. We use the BGP tables retrieved from RIS as the BGP tables from the *leaky neighbour* AS that are leaked by router L to A. When L advertises all its routes to A, only those corresponding to peers or providers of L are actual leaks. L's own routes and the routes from its customers are legitimate announcements to A. We identify all these routes using the relationships inferred by CAIDA to determine which neighbours of L are its customers and we remove them from the leak count. The remaining routes are considered to be leaks. We next compute the sensitivity of ASIRIA when detecting leaks in router A.

In average across all the 302 experiments (each one corresponding to the leakage of the routing table of a different RIS monitor), the ASIRIA-enabled router can detect 17.7% of the routes leaked by the neighbour. The distribution for the different experiments is shown in Fig. 6. We observe that there are important differences between experiments regarding the leak detection capability. In the top 2% of

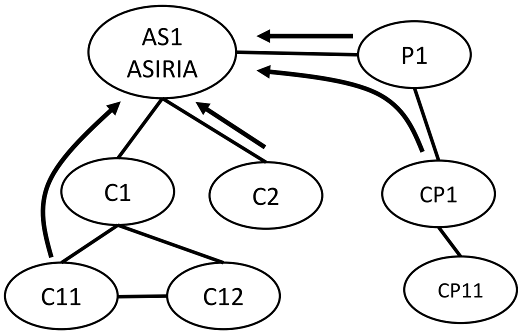

Fig. 4. Leak scenarios.

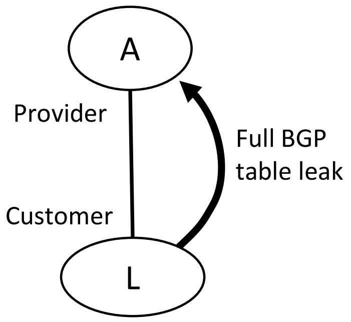

Fig. 5. Router leaking its routes to a provider.

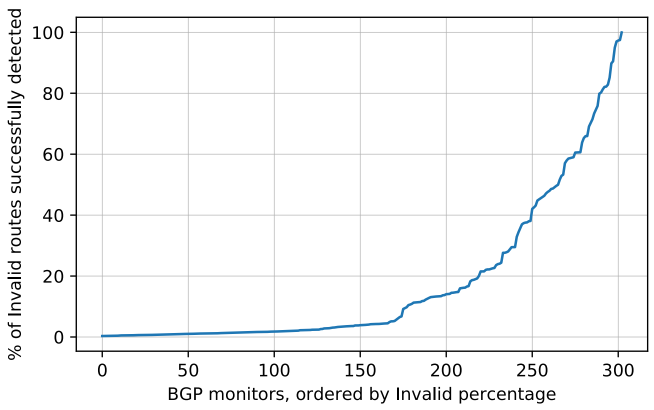

Fig. 6. Percentage of leaks detected in different experiments of the client AS leaking its full routing table to its provider. Different experiments account for different routing tables used from the leaky neighbour.

cases, ASIRIA is able to detect more than 90% of the routes as leaks, and in 20% of the cases ASIRIA can detect at least 40% of the leaks. In all these cases, ASIRIA provides immediate significant gains. On the other end, the bottom 30% of the cases only detect 1.6% of the leaks.

We mentioned that most of the information comes from the RIPE IRR and because of this, we expect that most of the relationships inferred involve European networks. Indeed, experiments using routing tables from monitors located in regions outside RIPE exhibit a much lower ratio of detected leaks. The ratio of leaks detected for the other regions are: USA: 2.7%, ASIA: 5.9%, LACN: 9.2% and AFR: 7.4%. As we assume that the provider running ASIRIA has information about the relationships with its peers, leaks that are undetected by the provider in this scenario are routes with AS paths that do not have any p2c nor p2p relationship tagged. Hence, these leaks remain undetected when propagated throughout the rest of the Internet. This means that leaks generated in regions where IRRs are less populated are more likely to go

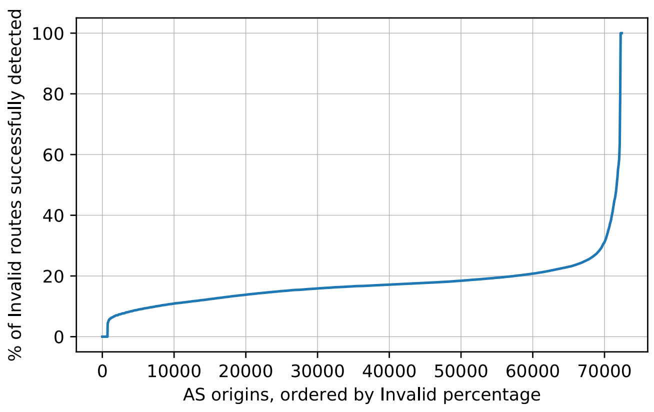

Fig. 7. Percentage of leaks detected in different experiments of all the client ASes leaking to a provider the routes for a single origin AS.

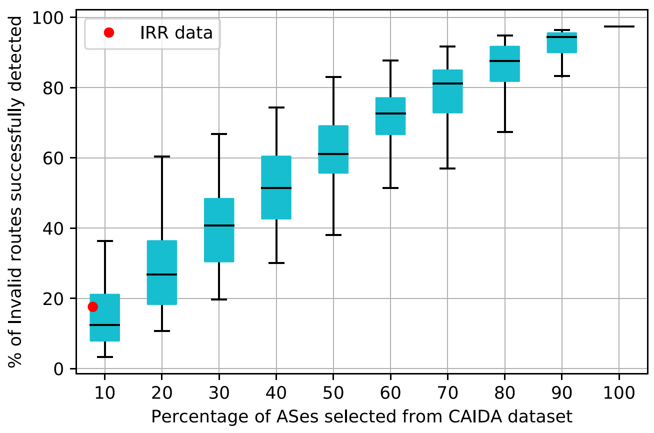

Fig. 8. Boxplot with the percentage of leaks detected when a client AS leaks its full routing table to a provider, using the relationships inferred by CAIDA to perform route validation. The *x*-axis represents the variation in the fraction of relationships available for route validation, with 100 random sets generated for each case. The result for route validation with IRR data is also depicted.

undetected and propagate globally. Besides, the impact of a leak is more severe closer to its source, so ASes in regions with more information in the IRR are less likely to experience severe disruptions caused by leaks than ASes in regions with less populated IRRs when deploying ASIRIA.

The analysis presented shows the case in which a direct neighbour of an ASIRIA router originates the leak. However, these results apply to any leak coming through a customer AS or a peering AS. This means that ASIRIA detects leaks generated by an AS within the customer's cone of the detecting AS or within the customer cone of the peer of the detecting AS, even though none of the intermediate routers are applying ASIRIA. The reasoning is straightforward: a route received from the customer cone or the customer cone of a peer is identified as upstream, so any p2c/p2p relationship breaks the validity rules for such route. Thus, the previous figures also represent the capacity of an ASIRIA router to detect leaked paths if the leaker is located in its customer cone or in the peer's customer cone, as long as the route is visible to the ASIRIA router.

We next look into the performance of ASIRIA detecting leaks in case the neighbour leaks the routes originated from a specific origin AS. The results are presented in Fig. 7. When we the leakage is limited to the routes of a specific origin AS, instead of leaking the full routing table as we considered in the previous analysis, the median percentage of routes detected by ASIRIA is 16.7%.

5.4. Variation of the sensitivity of ASIRIA with the number of annotated relationships

In the previous section, we observe that in some cases ASIRIA fails to detect a significant number of leaks. This is because there are many relationships between ASes for which there is no information in the IRR. In this section, we analyse how ASIRIA's performance improves as the number of relationships declared in the IRR increases.

We compute the evolution of the leak detection sensitivity of ASIRIA as a function of the number of AS relationships declared in the IRR. For this purpose, we leverage on the information about the p2c/p2p relationships provided by CAIDA, and we use it to compute ASIRIA's route leak sensitivity. We perform 10 batches of 100 experiments each. For each of the batches, we include an extra 10% of the information about AS relationships available in CAIDA, i.e., for the first experiments we only include the information about 10% of ASes from the CAIDA dataset, 20% for the second batch and so on. All the experiments within a batch have the same percentage of added ASes but the specific ASes added are randomly selected. Then we use the same methodology described in the previous section: we compute the fraction of detected route leaks for each experiment. This allow us to gain an insight of how ASIRIA's route leak sensitivity could evolve as the IRR gets populated.

The results of the fraction of routes identified as leaks are shown in Fig. 8. We include in the graph the value for the performance of ASIRIA when using the relationships inferred from the IRR, presented in Section 5.3, i.e., the 17.6% of the leaked routes are detected. The position in the X axis represents the fraction of relationships observed in the BGP routes to be validated that are present in the annotated IRR dataset, 7.9%. Fig. 8 indicates that almost all routes are identified as leaks when the complete CAIDA relationship dataset is used. This is because the whole majority of the relationships for the routes we validate are annotated in this dataset, as these are part of the BGP tables used as input to the CAIDA dataset inference process [11]. Note that this large overlap is not expected for an arbitrary router (different to a RIS monitor) using the CAIDA dataset to validate the routes it receives. The mean value for the case in which only the 10% of the relationships observed in the CAIDA dataset are used to validate the routes is 14.7%. In general, the fraction of Invalid routes identified by the ASIRIA routers is higher than the fraction of ASes disclosing its relationships, except for the case in which all ASes make its information public. The performance of the IRR dataset is consistent with the performance of the CAIDA dataset one for a similar number of relationships disclosed.

In the previous analysis, we randomly selected the information included in the IRR for our computations of the evolution of ASIRIA's route leak sensitivity. We next look into the diversity of the impact on ASIRIA's performance depending on which ASes declare their relationships. That is, we look whether there is a small set of ASes that, if they declared their relationships, ASIRIA's performance would improve significantly.

We compute the frequency in which the pairs of ASes are present in the AS path of the BGP routes present in the RIS routing tables. We identify all the AS pairs that appear in the AS path of every route and were not registered in the IRR, and we count the number of occurrences of each pair. Then, we order the AS pairs by decreasing frequency. Thus, the first AS pair has the potential (if annotated as c2p or p2p) to invalidate the larger number of routes, compared to the annotation of the relationships other less frequent AS pairs, etc.

We present in Fig. 9 the number of routes identified as leaks as each AS pair is being annotated. There are 363K different pairs for 26.2M routes. The distribution is highly skewed, as annotating a few pairs results in a large fraction of the routes being tagged as Invalid. For example, the first AS pair results in 1.5% of invalid routes, the top 10 pairs in 9.6% and the top 1000 pairs in 83.6% (i.e., annotating a 0.28% of the pairs allow detecting 83.6% of the leaked routes). The most frequent pairs appearing in the routes correspond mostly to

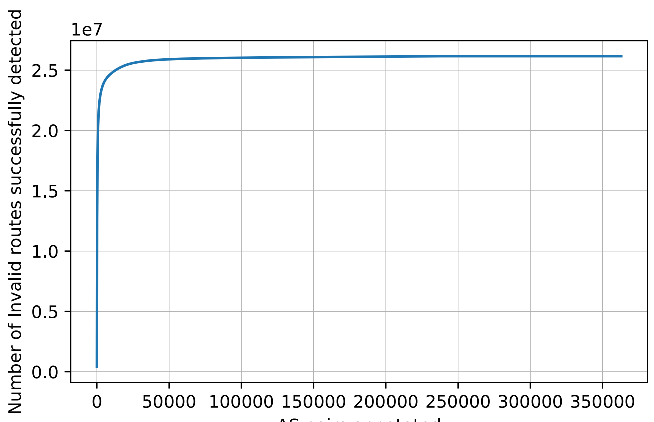

Fig. 9. Variation of the number of Invalid routes detected as AS pairs are annotated in the IRR in the order of decreasing frequency.

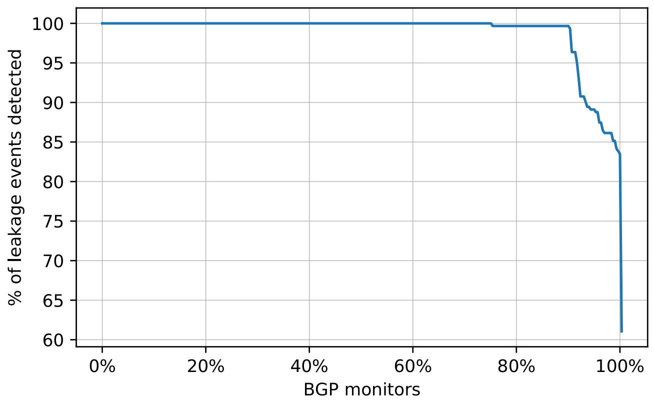

Fig. 10. Distribution of the number of leakage events detected using ASIRIA across the different experiments performed.

interconnections between ASes up in the routing hierarchy: in the top 20 pairs, we observe 23 occurrences of ASes in the top 10 of CAIDA's AS rank [17]. In this ranking, ASes ordered by their customer cone size, i.e., the number of their direct and indirect customers. This indicates that promoting the annotation of the interconnections between large providers would provide a large return in terms of the ability to detect routing leaks.

5.5. Sensitivity of the ASIRIA leakage event alarm system

In addition to detecting and marking individual route leaks, ASIRIA can also detect leakage events and trigger alarms that can alert the network administrator of an ongoing leakage event. An ASIRIA alarm is triggered when the number of leaks present in the BGP routing table of the ASIRIA enabled router increments beyond a configured threshold **Th** within a short period of time **p**, i.e., at least a **Th** increase in **p** minutes or less.

In order to set a value for **Th**, we compute the number of invalid paths present in the analysed BGP routing tables when there were no reports of any leakage event (according to BGPstream) and compare it to the number of paths identified as invalid when simulating a leak of a full BGP table from the neighbour in the scenario depicted in Fig. 5. We find that the mean number of invalid paths per BGP routing table in steady state is 96 (using the data from Section 5.2), while the mean number of invalid paths detected after a leakage of a full routing table is 18,454 (using the data from Section 5.3). Setting **Th** as closer as possible to the value of invalid paths present in steady state would allow ASIRIA to detect leakage events that involve few routes, but it may also result in a high number of false alarms. However, given the very large gap between the number of invalid paths in steady state and the number of invalid paths in the case of a full BGP feed, we set **Th** to a 100% increase in order to be conservative.

We next compute the sensitivity of the ASIRIA alarm system emulating scenarios involving several offending ASes and several victims. As before, we consider the scenario depicted in Fig. 5. We compute the number of false positives (using the methodology described in Section 5.2) and detected leaks (using the methodology described in Section 5.3) for the different combinations of offending AS and ISP. Thus, to perform the emulation of the different scenarios, we use all possible combinations of two RIS monitors, and we use one as the offending, i.e., the client AS leaking its full routing table, and the other as being the victim provider AS that receives the leaked routes. For each combination, we compute if the ASIRIA alarm system using the configured threshold (100% increase in the number of leaks) would indeed detect the leakage event.

Fig. 10 plots the distribution of the number of leakage events detected by the ISPs running ASIRIA. We can see that 74% of the ISPs detect 100% of the leakage events, 90% of the ISPs detect at least 99% of the leakage events and 99.9% of the ISPs detect at least 83% of the leakage events.

We next look into the temporal dimension of the leakage event, and set the duration of the time bin **p** used in ASIRIA to detect them. According to [18], it takes up to 60 s to receive and process a full routing table from a BGP peer in normal conditions. When the leak is originated farther away, the announcements of the different paths will be more separated in time from each other. According to [19], while measuring the effects of BGP announcements in a large number of vantage points, the routes resulting from an announce of a prefix can be separated up to 150 s, while the routes resulting from the withdraw of a prefix can be separated up to 240 s. In order to account for that, and enable ASIRIA to detect remote leakage events, we set **p** to 300 s (i.e., 5 min).

We verify if the proposed time interval **p** triggers false alarms when used with the proposed threshold **Th**. We analyse the BGP feed of the 302 monitors in the interval from 10:00 AM until 4:00 PM on the 2021-03-01. No leakage events were reported during this period according to BGPStream. Then, for every 5-minute bin of the analysed period, we compute the increase in the number of invalid paths as a ratio of the invalid paths present in steady state in the BGP table of each monitor. The maximum increase observed across all the monitors during this period was of 0.2%, far from the 100% defined for the threshold **Th**. The conservative selection of the value for the threshold allows ASIRIA to avoid false alarms.

With all this, we set **Th** to 100% and **p** to 5 min, the ASIRIA leakage event alarm system allows 90% of the monitors to detect at least 99% of leakage events generated by a neighbour.

5.6. Route validation execution time

We finally discuss the performance of the route validation process, to assess that it could be integrated in real-time route processing. The mean time required to validate a route with the Python code used for our evaluation is 1.6 microseconds in a single threaded code (Intel Core i7-5600U CPU, 2.60 GHz). This means that about 625,000 routes can be validated per second. Considering that the mean number of BGP updates per day was 50,000 during 2021 [20] and that every BGP update carries one route at most (or none, for messages only withdrawing routes), the whole number of updates for a day can be processed in less than a second. In case that the whole routing table is advertised (e.g., when a BGP session is established or when a full BGP table is leaked), the validation of all the routes involved would take 1.5 s (as per [21] at the end of 2021, the full BGP table accounts for around 900,000 prefixes [21]). These datapoints suggest that validation rate is enough for practical purposes.

If required, faster route validation could be achieved by implementing the validation logic in a lower-level language, or by parallelising the validation process (note that the relationships data used as input for the validation process is quite stable for the time scale of the validation process, so that consistency would be easy to achieve to support parallelisation).

6. Combining ASPA and ASIRIA

Autonomous System Provider Authorisation (ASPA) [22] is a proposal for a new RPKI extension to protect from route leaks. ASPA defines a new object for the RPKI that enables an AS to declare its Customer-Provider relationships with neighbouring ASes. ASPA also defines a procedure by which ASPA objects are used to detect leaks by inspecting the AS-path information present in BGP Update messages. The BGP route selection process is modified so that invalid routes are discarded.

As many other novel security mechanisms, ASPA faces the challenge of providing value to early adopters. The protection provided by ASPA is directly related to the number of ASPA objects present in the RPKI. During the early phases of adoption there are few ASPA objects in the RPKI, hence the resulting protection will be very limited. This in turn generates little incentives for ASes to deploy ASPA and validate the routes. If we take the deployment of the RPKI as a reference, according to Testart et al. [15], seven years after the completion of the standardisation of the RPKI only 17% of the routed prefixes were registered in the RPKI, and only 3% of a set of analysed provider networks enforced the route validation rules. Thus, surmounting the difficulty created by the interdependency between utility and deployment of the proposed solution is critical for gaining widespread adoption of ASPA and hence obtaining Internet-wide protection against route leaks. ASIRIA can help surmounting this challenge and foster the adoption of leak protection mechanisms such as ASPA.

The fundamental difference between ASPA and ASIRIA is the nature of the information used regarding AS relationships. ASPA uses a new RPKI object specifically designed for this, while ASIRIA infers AS relationships from the IRR. While we can expect ASPA information to be more reliable and accurate when it becomes available, ASIRIA can rely on existing information, thus providing value to early adopters. We believe that both solutions can be combined, so that ASIRIA can be used while ASPA objects are scarce. Then, some c2p relationships will be obtained from the RPKI and others from the IRR. In this way, ASIRIA can help bootstrapping ASPA.

6.1. ASPA overview

In ASPA, ASes can declare their (direct) transit providers in the RPKI [1], using the newly defined ASPA record. An ASPA-enabled AS uses this information to verify AS paths of the received routes and detect leaks in real time.

An ASPA record is a digitally signed object where the signing AS lists the set of providers authorised to propagate its routes upstream to other providers. It is expressed as ASPA(AS1, AFI, [AS2, AS3, …]), representing that AS1 declares that AS2, AS3, … are all its providers for the AFI Address Family.

ASPA defines a route verification process that allows an ASPA-enabled router to determine whether the status of a route is Valid, Invalid, Unknown or Unverifiable by analysing the AS path attribute of the route. An Invalid path is a path for which a match with any of the sequences described as a leak in Section 2 has been determined. An Unknown path has AS pairs for which there is no relationship information, so that the validity check cannot be fully performed. Unverifiable paths contain at least an AS_SET segment, as a result of BGP aggregation performed along the propagation path. The remaining paths are Valid ones.

ASPA recommends discarding Invalid and Unverifiable routes and accepting routes marked as Unknown, enabling ASPA use in scenarios in which the mechanism has not been fully deployed. The outcome of ASPA is accepting or rejecting the route, but it does not affect the preferences of the accepted routes.

6.2. ASIRIA route verification process and its integration with ASPA

We adapt the ASIRIA route verification to be a plug-in into the ASPA route verification process.

When a router receives a BGP route, it starts by executing the route verification process defined for ASPA. It extracts the pairs of ASes included sequentially in the AS_PATH attribute of the route. To detect route leaks, the router looks for registered relationships for the AS pairs at the AS_PATH. It first searches for a match in the ASPA DB and only if the relationship is Unknown by ASPA, it searches for a match with the information available in the ASIRIA DB. This means that the information in the ASPA DB takes precedence over the information available in the ASIRIA DB. The semantics of the information included in the ASIRIA DB is not the same as for the ASPA DB. The main difference is that when an AS issues an ASPA record, it indicates all the providers of the AS, and thus, allows ASPA to conclude that all ASes that are not included in the AS' ASPA record are not its providers. This means that once an AS issues an ASPA record, all providers must be enumerated. In the case of ASIRIA, ASIRIA is able to identify c2p/p2c relationships with a high degree of confidence but the lack of information about the relationship between a pair of ASes in the ASIRIA DB does not necessarily implies that there is no relationship between the pair of ASes. This results in a slightly different validation process for ASPA and ASIRIA.

The router takes different actions depending on if the path is determined as Invalid using solely ASPA information or if ASIRIA was also used. In the former case, the route can be discarded (as per ASPA recommendation) while the in latter case the preference of the route can be decreased, so alternative valid routes can be preferred.

7. Related work

The literature regarding BGP security is vast. In this section we focus on other work strictly related to BGP route leak protection, which is considerably more reduced, and we skip work related to other aspects of BGP security such as mechanisms to protect against hijacks.

The current practice to protect from route leaks is the configuration of filters [6]. An ISP should configure inbound filters in the BGP sessions with its peers and customers, so that only routes containing the prefixes and/or AS numbers allocated to its peers/customers and its peer/customers' customers are accepted while all other routes are filtered out. Frequently, these filters are built using the information available in the Internet Routing Registries (IRRs). In order to enable that, the ISP's customers and peers should populate the IRR with the information regarding their prefixes and AS numbers and they should also require their own customers to do so. However, the large number of route hijack and leak events indicate either lack of incentives or difficulties in correctly deploying such measures. One reason for not deploying these measures may be the high impact of an error in the configuration, which may result in filtering routes and hence traffic from its (paying) clients. These configurations are complex due to the high number of relationships and the dynamic nature of routing information. The challenge of keeping accurate configurations grows for large ISPs with multi-tier customer cones. ASIRIA is an alternative that provides route leak protection when filtering solutions do not suit the ISPs operating requirements.

Azimov et al. [23] extend the BGP OPEN message to explicitly include information about the type of relationship (peer, customer, provider, route server, route server client) between neighbour ASes. Thus, neighbouring ASes can verify and enforce that there is mutual agreement on the type of relationship and that the BGP configuration is consistent with the type of relationship. This information by itself it is not enough to provide protection against route leaks, as it does not allow to determine whether the route conforms to any of the leak patterns presented in Section 2. However, such extension complements the ASIRIA solution, since ASIRIA relies on local information about the type of relationship an AS has with its neighbouring ASes. Having it explicitly announced in the BGP OPEN message would certainly make ASIRIA more robust by avoiding mismatches between the configuration of neighbouring routers, or between the actual local router configuration and ASIRIA's view.

Azimov et al. [23] also defines a new, optional, transitive, Only to Customer (OTC) attribute to mark routes that must only be announced downwards to customers. Tagging routes with this attribute allows ASes to detect when the route is leaking and filter it. The approach based on the OTC attribute has several merits, including limited disclosure of routing policy information, which in some cases is considered strategic information. The main advantage that ASIRIA exhibits compared to the OTC approach is regarding initial deployment, as ASIRIA relies on existing information in the IRR. So, while early OTC adopters do not benefit from any protection, the first ASes deploying ASIRIA would obtain the protection described in Section 5, fostering its adoption.

Cohen et al. [24] define a new *path-end record* for the RPKI which includes a "non-transit indicator" (NTI) flag. This flag allows ASes to state (in the RPKI) that they are stub ASes and they should only appear at the end of the AS paths in routes announced in BGP. In this way, it prevents the AS setting the flag to generate route leaks. This is an elegant approach that is part of a more general solution to provide BGP security. However, the protection of NTI is limited to route leaks generated by stub ASes, while ASIRIA can protect also from leaks generated by other ASes, including any form of customer and peer combinations. Besides, NTI requires populating the RPKI with the new record, while ASIRIA leverages on existing information in the IRR. Also, while it seems apparent that NTI contributes to protect against route leaks caused by errors, it is unclear how effective they would be against malicious routing leaks, since it is unlikely an AS that intents to maliciously inject a route leak will set the NTI flag. ASIRIA, on the other hand, can detect leaked routes using the information about the relationships with adjacent ASes, even if the leaker does not collaborate.

There are third party services such as BGPmon[4] and Qrator[5] that detect global route leaks and inform subscribed ASes of such events. The main differences between ASIRIA and those systems are that ASIRIA runs locally, while BGPmon and Qrator are external services. In cases of major disruptions that affect the connectivity, access to third party services may also be compromised. In addition, BGPmon and Qrator use algorithmic inference of relationships, while ASIRIA uses the information declared by ASes in the IRR. This means that each AS has direct control over the information used for inferring relationships that it is involved in and if an AS detects that one of its relationships is wrongly inferred, it can directly change the IRR and potentially correct it. Such modification is not straightforward in the case of algorithmic inference of relationships.

Similarly to our work, [25] proposes to identify route leaks by detecting non valley-free paths. However, while we identify non valley-free paths using IRR information, [25] does it by observing the BGP path dynamics. In particular, they infer routing policies by identifying *safe patterns* from long-lived paths and concurrency of non-complying paths. One main difference between the two approaches is that with ASIRIA the involved ISPs have control over the data used by leak detection mechanism (i.e., their policy declaration on the IRR) and in case of an error/problem leading to a miss-inference, the ISPs can correct it.

Peerlock [26] is a leak protection mechanism currently used by large Internet transit providers. Peerlock is deployed between two peering ASes, one of them being the protected AS and the other the protector one. The protector AS configures its filters to drop any route involving the protected AS prefixes that is not announced by the protected AS or its designated upstream providers. ASIRIA has a few advantages compared to Peerlock. First, Peerlock requires pairwise (manual) coordination between all the involved ASes, which is cumbersome; while it can work for a limited number of ASes, it does not scale well Internet-wide. Also, Peerlock protection scope is limited to the first or second ASes in the AS path, while ASIRIA can protect all the relationships along the AS path. Peerlock has been designed by and for large transit carriers, notably Tier 1 providers, so in that context, the limitations are acceptable. ASIRIA on the other hand, is targeting universal deployment.

Regarding the inference of AS relationships, there is also a vast amount of related work. In order to be succinct, we only cover the related work that used IRR information to infer AS relationships. [27] uses the IRR information to infer peering relationships between ASes. In this context, peering is defined as two ASes which have a BGP session between them. [27] does not attempt to use IRR information to infer the type of relationship, which is the goal of ASIRIA when analysing the IRR data. [28] uses IRR information to infer the type of relationship between ASes, but unfortunately, they do not specify the algorithm used for this process (the paper focusses on how to combine multiple data sources to have a more complete view of the relationships inferred).

8. Conclusions

We presented ASIRIA, a route leak detection and protection mechanism based on the information available at the IRRs. ASIRIA defines a method to infer the AS relationships from the information included by the ASes themselves in the registry. With this information, it validates the AS path of the received BGP routes, identifying as leaks those that violate usual business rules. Leak routes are deprioritized over other routes and large variations in the number of detected leaks trigger an alarm indicating that a close AS may generating a leakage event.

The merits of this approach compared to existing ones include benefits for early adopters through the use existing information (routing policy records at the IRR), the ability of the ASes to control the information about their own relationship used by ASIRIA, and a less aggressive strategy compared to accept-list filter approaches, as only explicitly identified leaked routes are affected and these routes can still be used if no alternatives exist.

We next enumerate the most important characteristics for 75 network operators regarding BGP security incident mitigation as presented in a survey done in 2018 [29], and we expand on how ASIRIA performs from the perspective of each of the top four features.

(1) **Effectiveness of mitigation**. Unsurprisingly, the most important feature for ISPs is the effectiveness of the mechanism. Due to the distributed nature of the problem, the effectiveness of any solution depends on its degree of adoption across the Internet,

[4] https://www.bgpmon.net/.
[5] https://radar.qrator.net/.

usually providing little or no value for early adopters. ASIRIA relies on existing information in the IRR to infer relationships. Because of this, it can protect early adopters. As presented, ASIRIA allows 90% of the analysed ASes to detect more than 99% of the leakage events. In addition, ASIRIA allows early adopters to detect 17% of leaked routes in average. The effectiveness of ASIRIA will increase as more ASes populate the IRR, which is actively being promoted by initiatives such as MANRS. In the survey, 80% of the queried ISPs were willing to disclose their routing policies to increase BGP security.

(2) **Fast mitigation**. A router running ASIRIA has the information required to validate the paths locally available, so invalid paths are detected upon its reception as part of the BGP route selection process. In the case of leakage event detection, we have set the detection mechanism to use a 5-minute period, but if the number of invalid paths reached the configured threshold **Th** earlier, the detection occurs faster. This is likely to be the case of major route leak events, when many paths are affected.

(3) **Self-managed/operated**. 61% of the ISPs in the survey declared to rely on notifications from third-party services (e.g., BGPmon and Qrator) to detect BGP route hijacking incidents, and another 61% of ISPs declared that they would rather avoid outsourcing these kinds of functions. A distinctive characteristic of ASIRIA is that it runs locally, and it is not a service provided by a third party. It only relies on public IRR information that is periodically updated, but no real time access to the IRR is required. ASIRIA is able to provide effectiveness to early providers, a feature typically only available to solutions provided by third parties, while being locally operated.

(4) **Ease of operating/troubleshooting**. As 61% of ISPs in the survey already use third party notification of relevant events, the ASIRIA route leak alarm may fit in their existing operational workflow, although in this case the service could be operated in-house. ASIRIA detects both leakage events and specific invalid routes. By marking specific routes as invalid, ASIRIA not only avoids those routes, but also provides information to the network administrator to troubleshoot and determine the offending AS causing the route leak.

As future work we intend to extend ASIRIA to use Machine Learning (ML) algorithms to detect leaks using the partial information present in the IRR. The ML system would take as input the routes detected as leaks by the current version of ASIRIA, and it could be used to infer whether other unmarked routes are potential leaks, improving the overall ASIRIA performance. The ML-based inference should only be applied to relationships not explicitly annotated by the ASes in the IRR, to preserve the ability of ASes to retain full control of the detection process over routes containing their relationships.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by Huawei through the Internet Routing Blockchain project, by the EU through the NGI Atlantic MCCA project and the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M21), and in the context of the V PRICIT (Regional Programme of Research and Technological Innovation).

References

[1] Matt Lepinski, Stephen Kent, An infrastructure to support secure internet routing, 2012, RFC 6480.
[2] Matt Lepinski, Kotikalapudi Sriram, BGPsec protocol specification, 2017, RFC 8205.
[3] Andree Toonk, Massive route leak causes internet slowdown, 2015, https://www.bgpmon.net/massive-route-leak-cause-internet-slowdown/.
[4] J. Fleury, et al., Anatomy of a route leak, 2019, https://www.lacnic.net/innovaportal/file/4016/1/cloudflare---anatomy-of-a-route-leak.pdf.
[5] Kotikalapudi Sriram, Doug Montgomery, Danny R. McPherson, Eric Osterweil, Brian Dickson, Problem definition and classification of BGP route leaks, 2016, RFC 7908.
[6] MANRS, MANRS implementation guide, filtering, 2020, https://www.manrs.org/isps/guide/filtering/. (Accessed: 11 November 2020).
[7] Lixin Gao, On inferring autonomous system relationships in the internet, IEEE/ACM Trans. Netw. 9 (6) (2001) 733–745.
[8] Peyman Faratin, David D Clark, Steven Bauer, William Lehr, Patrick W Gilmore, Arthur Berger, The growing complexity of internet interconnection, Commun. Strateg. (72) (2008) 51.
[9] V. Giotsas, M. Luckie, B. Huffaker, k. claffy, Inferring complex AS relationships, in: ACM Internet Measurement Conference, IMC, 2014, pp. 23–30.
[10] Randy Bush, Rob Austein, The resource public key infrastructure (RPKI) to router protocol, version 1, 2017, RFC 8210.
[11] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, KC Claffy, AS relationships, customer cones, and validation, in: Proceedings of the 2013 Conference on Internet Measurement Conference, 2013, pp. 243–256.
[12] David Kessens, Tony J. Bates, Cengiz Alaettinoglu, David Meyer, Curtis Villamizar, Marten Terpstra, Daniel Karrenberg, Elise P. Gerich, Routing policy specification language (RPSL), 1999, RFC 2622.
[13] David Meyer, Cengiz Alaettinoglu, Dr. Carol Orange, Mark R. Prior, Dr. Joachim Schmitz, Using RPSL in practice, 1999, RFC 2650.
[14] Brenden Kuerbis, Milton Mueller, Internet routing registries, data governance, and security, J. Cyber Policy 2 (1) (2017) 64–81.
[15] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, David Clark, To filter or not to filter: Measuring the benefits of registering in the RPKI today, in: International Conference on Passive and Active Network Measurement, Springer, 2020, pp. 71–87.
[16] Vasileios Giotsas, Shi Zhou, Valley-free violation in internet routing - analysis based on bgp community data, in: 2012 IEEE International Conference on Communications, ICC, 2012, pp. 1193–1197.
[17] CAIDA, AS Rank: A ranking of the largest autonomous systems in the internet, 2021, https://asrank.caida.org/. (Accessed: 30 January 2022).
[18] Juan Brenes, Alberto García-Martínez, Marcelo Bagnulo, Andra Lutu, Cristel Pelsser, Power prefixes prioritization for smarter BGP reconvergence, IEEE/ACM Trans. Netw. 28 (3) (2020) 1074–1087.
[19] Alberto García-Martínez, Marcelo Bagnulo, Measuring BGP route propagation times, IEEE Commun. Lett. 23 (12) (2019) 2432–2436.
[20] Geoff Huston, BGP In 2021 - BGP updates, 2022, https://blog.apnic.net/2022/01/13/bgp-updates-2021/. (Accessed: 30 January 2022).
[21] Geoff Huston, BGP In 2021 - The BGP table, 2022, https://blog.apnic.net/2022/01/06/bgp-in-2021-the-bgp-table/. (Accessed: 30 January 2022).
[22] Alexander Azimov, Eugene Bogomazov, Randy Bush, Keyur Patel, Job Snijders, Verification of AS_PATH Using the Resource Certificate Public Key Infrastructure and Autonomous System Provider Authorization, Internet-Draft draft-ietf-sidrops-aspa-verification-08, Internet Engineering Task Force, 2021, Work in Progress.
[23] Alexander Azimov, Eugene Bogomazov, Randy Bush, Keyur Patel, Kotikalapudi Sriram, Route Leak Prevention using Roles in Update and Open messages, Internet-Draf draft-ietf-idr-bgp-open-policy-16, Internet Engineering Task Force, 2021, Work in Progress.
[24] Avichai Cohen, Yossi Gilad, Amir Herzberg, Michael Schapira, Jumpstarting BGP security with path-end validation, in: ACM SIGCOMM Conference, New York, NY, USA, 2016, p. 342 355.
[25] Shen Su, Beichuan Zhang, Lin Ye, Hongli Zhang, Nathan Yee, Towards real-time route leak events detection, in: 2015 IEEE International Conference on Communications, ICC, 2015, pp. 7192–7197.
[26] Tyler McDaniel, Jared M. Smith, Max Schuchard, Flexsealing BGP against route leaks: Peerlock active measurement and analysis, 2020.
[27] Giuseppe Di Battista, Tiziana Refice, Massimo Rimondini, How to extract BGP peering information from the internet routing registry, in: ACM SIGCOMM MineNet Workshop, 2006.
[28] Jianhong Xia, Lixin Gao, On the evaluation of AS relationship inferences [internet reachability/traffic flow applications], in: IEEE Global Telecommunications Conference, 2004. GLOBECOM '04, Vol. 3, 2004, pp. 1373–1377.

[29] Pavlos Sermpezis, Vasileios Kotronis, Alberto Dainotti, Xenofontas Dimitropoulos, A survey among network operators on BGP prefix hijacking, SIGCOMM Comput. Commun. Rev. 48 (1) (2018) 64–69.

Marcelo Bagnulo received the Electrical Engineering degree from the University of Uruguay and the Ph.D. degree in telecommunications from the Universidad Carlos III de Madrid (UC3M), Spain. Since 2008, he has been a tenured Associate Professor at UC3M. He has published more than 80 articles in the field of advanced communications in journals and congresses, including IEEE INFOCOM, ACM SIGCOMM, ACM Mobicom, ACM IMC, and IEEE/ACM TRANSACTIONS ON NETWORKING. He is the author of 19 RFCs in the Internet Engineering Task Force (IETF), including the Shim6 protocol for IPv6 multihoming and the NAT64/DNS64 tools suite for IPv6 transition. He has 26 Hindex and 4258 total citations. His research interests include Internet architecture and protocols, interdomain routing, and security. From 2009 to 2011, he was a member of the Internet Architecture Board.

Alberto García-Martínez received the degree in telecommunication engineering, in 1995, and the Ph.D. degree in telecommunications, in 1999. In 1998, he joined the Universidad Carlos III de Madrid (UC3M), where he has been an Associate Professor, since 2001. He has published more than 50 articles in technical journals (IEEE/ACM Transactions on Networking, Computer Networks), magazines (IEEE Wireless Communications, IEEE Communications Magazine), and conferences. He has coauthored three RFCs. His main interests include interdomain routing, transport protocols, network security, and blockchain technologies.

Stefano Angieri received a computer science bachelor degree in 2014 and a master degree in 2018 at Universitá degli studi Federico II di Napoli with a master thesis on blockchain technology. In 2018 he joined Universidad Carlos III de Madrid (UC3M) as Ph.D student. His main interest area is blockchain technology and routing security.

Andra Lutu is a Senior Researcher at Telefonica Research, in Madrid, Spain. Her main research interests lie in the areas of network measurements, interdomain routing and mobile networks. As part of Telefonica Research, Andra has been the recipient of an H2020 MSCA Individual Fellowship grant funding her work on Dynamic Interconnections for the Cellular Ecosystem (DICE).

Jinze Yang received his B.Eng. degree in Internet of Things Engineering and the Ph.D. degree in Electronic Engineering from the Queen Mary University of London (QMUL) in 2015 and 2020, respectively. He joined the Huawei Technologies Ltd., China in 2020. He is currently a senior engineer in Huawei Ltd. His research interests include data centre networks, Internet of things networks and information-centric networks.