A hybrid system for automatic exchanges of routing information

Author

STAMATIOS MARITSAS
stamatios.maritsas@os3.nl

supervised by

STAVROS KONSTANTARAS
s.konstantaras@uva.nl

GEORGE THESSALONIKEFS
george@nlnetlabs.nl

UNIVERSITY OF AMSTERDAM
Informatics Institute
Master of System and Network Engineering

DECEMBER 2016

Abstract

The exchange of routing information for BGP configurations is a critical functionality that help autonomous systems communicate with each other in an efficient and robust way. The initial effort for this exchange to be applied was through centralization, a concept that survived throughout the years and remains as legacy.

In this work, we propose a hybrid system for automatic exchange of routing information. Our approach proves that decentralization of policies can take place. It addresses security and benefits from using a hybrid model for achieving policy routing information exchange in an efficient way. Each administrative domain, in this system, is in charge of sharing, storing and updating its own policy information. We offer a new perspective and direction for discussions on routing information exchange. We focus on two aspects that are of paramount importance; decentralization of routing policies and security.

TABLE OF CONTENTS

List of Tables

List of Figures

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to the whole team of SNE for helping me select this research project. A special gratitude I give to my supervisors, Stavros Konstantaras and George Thessalonikefs (NLnet Labs), for guiding me through the whole procedure of my research project and answering all my questions and concerns. I would also like to thank Marijke Kaat (SURFnet) for her invaluable advice whenever it was needed.

CHAPTER 1

INTRODUCTION

The Internet, as a universal system, is a network-of-networks working in a collaborative way. Although, there are networks that may have no formal relationship, the traffic originated from one will be transmitted to another due to this collaboration. These networks end up in long prefix-lists and are getting announced via the Border Gateway Protocol (BGP). BGP is the only routing protocol in use for interconnection of networks or autonomous systems (ASes). It is the glue that holds all networks around the world together and robustness of the Internet is directly related to the robustness of BGP routing.

On the operational level it is getting the support of another technology, the Internet Routing Registry (IRR). IRR is a distributed set of databases, and part of its role is the storage of ASes' routing policies. The most important part of the IRR system are the so-called Regional Internet Registries (RIRs). RIRs are five in total and each of them is responsible for the allocation of the IP address space and AS numbers for a certain geographical region [1].

Network operators rely on a centralized concept for the exchanging of their policy information. They use many tools (e.g. IRRToolset, BGPq3, etc.) to retrieve the policy information from the IRR database system and use it for BGP configurations. However, this approach comes with some considerations.

According to McPherson et al. [2], one of the biggest weaknesses of the IRR system is that policy information is often out of date and inaccurate. The routing policy of many network operators may contain private peering agreements concerning, for example, private networks that they do not want to share with everybody. This may lead to inaccurate and outdated information, because network operators have privacy concerns and are not motivated enough to embrace and support this approach. Moreover, the policy exchange system, as it is now, lacks a proper authentication/authorization mechanism. The lack of authoritative IRR for resources does

not allow network operators to automatically know the authoritative IRR of a resource holder, which will contain their most up-to-date set of resources.

The goal of this project is to examine if it is possible to design a hybrid system that will automatically exchange routing policies between autonomous systems, in order to mitigate the above concerns. By designing a hybrid approach, control of policy information is transferred to authorities themselves. Every authority will be in charge of what policy information it shares with whom. In this way, network operators will have more incentives to keep their policy information properly updated and their privacy concerns will be alleviated.

Privacy concerns of network operators will be mitigated by providing a proper authentication/authorization model to this hybrid approach as well. We will focus on how we can authenticate/authorize both an originator and a requester of a routing policy.

1.1 Research questions

The main research question that arises is:

Is it possible to design a hybrid system to automatically exchange routing policies for BGP configurations?

To answer the main research question, the following sub-questions have been formed:

- Which would be the benefits of designing a hybrid approach? Would it be possible to alleviate the privacy concerns of the network operators?
- What is the potential of this hybrid system in terms of scalability and efficiency?
- What security aspects should this hybrid system employ?

1.2 Related work

In August 2015, Stella Vouteva and Tarcan Turgut, both alumni System and Network Engineering master students at University of Amsterdam, conducted a research on how we could automate the BGP configuration on edge routers using RPSL data files as input. First, they studied all existing BGP tools in order to find out which of these offered this required automation. Results of this research showed that current solutions are either too complex or outdated and most importantly not security-oriented. Subsequently, a comparison was performed and then they designed and implemented the so-called *BGPWizard* tool. A proof-of-concept that overcame the drawbacks of all the already existing tools. Their research covered the infrastructure part of a domain (from

the router to the IRR). The way to feed the tool with correct information in an alternative way is not yet covered (unexplored area)[3].

In addition, in 2014, Ralph Koning, Miroslav Živković, Stavros Konstantaras, Paola Grosso and Cees de Laat from University of Amsterdam (from SNE Research Group) in association with Farabi Iqbal from Delft University of Technology conducted a research on how we could automatically exchange XML files which include topology information in multi-domain environments. They proposed a hybrid system that consists of three entities. An index who holds pointers to topology providers, a topology provider who distributes the topology information, and a topology consumer who consumes data from both the index and the provider. In order to support their approach, they designed and implemented a proof-of-concept within the Automated GOLE (GLIF Open Lambda Exchange) environment. GOLE is part of the Global Lambda Integrated Facility (GLIF), which constitutes a worldwide programmable infrastructure. All network topologies were presented as a single XML document and the Automated GOLE used the standardized Network Markup Language (NML) format to describe these topologies [4].

Distributed Hash Tables (DHTs), such as Chord and Pastry, are another interesting part of related work. A DHT is a distributed approach that enables efficient key-based lookup of data in a peer-to-peer (P2P) overlay network. In other words, DHTs provide a mechanism to find a host responsible for a certain piece of data. Specifically, *(key, value)* pairs are stored in a DHT and any participant can efficiently look for the value associated with a given key. This scheme allows DHTs to scale to big numbers of nodes [5].

Freenet is also worth mentioning as it is a P2P platform via users can experience the freedom of speech on the Internet. It has a decentralized database that holds the information and a set of free software that gives the capability to users to publish and communicate on the Internet without fear of censorship. All nodes communicate with each other directly [6]. It uses a method called key-based routing (KBR) which is similar to DHTs. In DHTs we search for a host responsible for a specific piece of data, while in KBR we try to find the closest host for that data [7].

1.3 Outline

The rest of the report is organized as follows. We provide an overview of BGP along with an investigation concerning its security in Chapter 2. Additionally, IRR and RPSL technologies are presented along with some considerations. In Chapter 3 we present the methods that were used in order to obtain both theoretical and practical knowledge for the needs of this research.

We then present the design of our system in conjunction with the decisions that were made concerning its model and its security aspects in Chapter 4. Chapter 5 is completely dedicated to the architecture of our system. We summarize the components that constitute the system and we also describe the architecture operations. The scalability potential and the high availability

perspective of our system is discussed in Chapter 6 along with some implementation ideas.

Finally, we outline the conclusions of the conducted research In Chapter 7 and the report ends with our recommendations for future work in Chapter 8.

CHAPTER 2

BACKGROUND

2.1 Border Gateway Protocol

Internet is a global, decentralized network that consists of a huge number of smaller interconnected networks. Each of these networks is comprised of a number of end systems, referred to as hosts and a number of intermediate systems, called routers that form a so-called Autonomous System (AS). An autonomous system is an administrative domain that runs under its own administration and routing policies.

Routing protocols are used for communication between routers within the same domain or different domains. They can be summarized in two categories, intra-domain protocols or Interior Gateway Protocols (IGPs) and inter-domain protocols or Exterior Gateway Protocols (EGPs). IGPs aim to exchange routing information between devices within the same domain, while the EGPs are used to exchange routing and reachability information between devices that belong to different domains.

In the scope of this project we are focusing on Border Gateway Protocol (BGP) which is the *de facto* EGP routing protocol used for inter-domain routing on the Internet. BGP is counting back its days from 1989 when it was first standardized in RFC 1105 [8]. After 11 years, and after a number of modifications, it reached its newest standardization in RFC 4271 [9] (BGP4).

A neighbor or a so-called BGP session needs to be established first between two BGP speakers. Using *iBGP* (internal BGP), communication between BGP peers within the same AS can be achieved, while *eBGP* (external BGP) is being used for BGP peers that belong to different ASes. BGP is the only routing protocol that uses TCP for the establishment of its sessions (over port 179). In the following picture we illustrate an overview of the two BGP sessions:

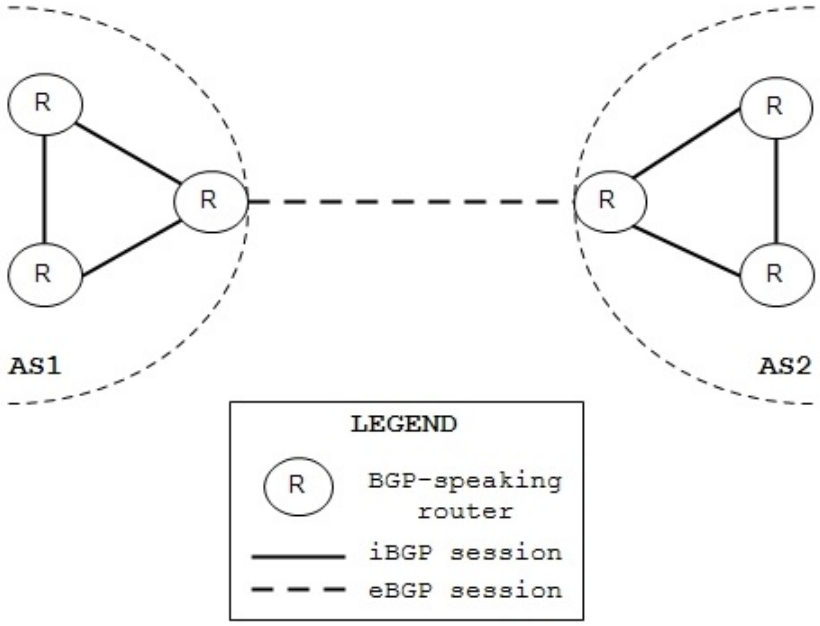

Figure 2.1: Overview of iBGP and eBGP sessions among BGP peers

BGP speakers exchange routes via UPDATE messages. Specifically, a BGP route associates a set of destination systems with the attributes of the path to those systems. The IP addresses of the destination systems are contained in an IP address prefix which is carried in the Network Layer Reachability Information (NLRI) field of an UPDATE message. Attributes, on the other hand, are used by the best route selection algorithm to determine the best path to the destination systems. Based on this information inside the UPDATE messages, which are described in detail in Section 5 of RFC 4271 [8], BGP-speaking routers can make policy decisions. Policies determine a set of rules on how routing and reachability information is exchanged and used between BGP routers.

2.1.1 BGP policies

BGP's key feature is that it allows network operators to define routing policies. A routing policy describes how a network is operated, containing information such as:

- whom does an AS connect with
- which route prefixes are announced to others
- which route prefixes are accepted from others
- what are the desired preferences, etc.

These policies are based on either *paid peering* or *settlement-free peering*. Internet access can be granted in *paid peering* only when a fee is involved, while in *settlement-free peering* the exchange of traffic is done for mutual benefit [10].

According to Vanbever et al. [11] and Caesar et al. [12], BGP policies can be classified into four main categories:

1. Transit policies
2. Traffic engineering policies
3. Scalability policies
4. Security-related policies

Transit policies

This kind of policies describe how routes are being propagated among ASes according to their established *business relationships*. The four common business relationships types that ASes establish are: *Provider-to-Customer*, *Customer-to-Provider*, *Peer-to-Peer* and *Sibling-to-Sibling* [13].

In the first two types a customer AS pays a fee to a provider AS in order to get transit access to the Internet. In a *peer-to-peer* business relationship, two ASes exchange their local routes without any payment involvement, while in a *sibling-to-sibling* relationship two ASes owned by the same network operator (different AS numbers) may advertise to each other all routes that they learn from their neighbors.

The goal of transit policies is twofold. First, they are used to influence the decision process of a BGP router (e.g. by manipulating the `LOCAL_PREF` attribute), which is the final phase where a router actually decides which path is going to be chosen. Second, they are used in controlling the route exporting (e.g. by using the `COMMUNITY` attribute) [12].

Traffic engineering policies

These policies are tightly coupled with performance. BGP routers often reach a point where a number of available routes is equally preferred and the reason for that is that many ASes are usually connected to several locations in order to reduce their delay and improve availability. This is where traffic engineering policies come into play. The goal of these policies is to direct the traffic in such a way that certain maximum performance criteria are met. Often this is done by modifying the import policies applied to the BGP routers [11] [12].

Scalability policies

As their name states, these policies are used for scalability reasons. There are many times that misconfigurations and human errors in neighboring ASes can lead to a massive number

of updates. This can overload the processing power or the memory of the routers, leading to malfunctioning and even failures. By establishing scalability policies, problems like the ones mentioned can be confronted and the flexibility of a network can be preserved.

Scalability policies usually aim at limiting the routing table size (by filtering and manipulating `COMMUNITY` attributes) and limiting the number of changes to the routes, which is usually done by suppressing flapping routes. Flap damping is a mechanism for BGP speaking routers that constrains routes that reach a large number of updates ("flapping") [12] [14].

Security-related policies

An AS is highly vulnerable to false BGP information announced by its neighbors. Misconfigurations may also lead to unstable or incorrect traffic forwarding as well. The security-related policies aim to limit and counterpart these problems. BGP security concerns and existing solutions will be discussed more in depth in Subsection 2.1.2. In this perspective, every self-respecting network operator has to defend its AS security as much as it can by establishing related policies. Some common goals of these policies are [12] [15]:

- Removing invalid routes (by import filtering)
- Security of network infrastructure (by export filtering)
- Protection of routing policies' integrity (by rewriting attributes)
- Blocking of Denial-of-Service (DoS) attacks (by filtering and damping)

2.1.2 BGP attacks and solutions

By the time of its creation and standardization in RFC 1105 [8] in the late 80's, BGP was a fairly simple, not security-oriented, path-vector protocol used to interconnect different ASes on the Internet. The number of networks at that time was smaller comparing to the huge number that exists today and the trust between network operators was more consistent and robust.

As the number of networks was growing, trust between ASes was declining and things got more complicated. Many modifications and mechanisms were added to make BGP more suitable to the incremental, day by day needs of ISPs, but its latest version (BGP4) still remains not security-oriented. Although this was necessary, it increased the complexity of the routing process.

Attacks

According to [16] we can classify BGP attacks in four categories: *modification attack*, *misconfiguration*, *exposing attack* and *contamination attack*. In *modification attack* a malicious AS modifies the `AS_PATH` attribute that is included in the BGP UPDATE message. Specifically, a malicious user changes a valid value of `AS_PATH` to an shorter and invalid one, enforcing in this way other

neighboring ASes to choose this fake path while updating their routing tables via the exchange of BGP UPDATE messages [17].

Mahajan et al. [15] conducted an extensive research on the various kinds of BGP *misconfigurations*. We interpret as misconfigurations all the configuration errors that result in an unintentional announcement (in case of an attack), or non-announcement of BGP prefixes. There are two types: *origin misconfiguration* and *export misconfiguration*. The first one takes place when an AS injects specific prefixes (even private) into the global BGP tables or announces other ASes' prefixes (e.g. YouTube hijack incident by Pakistan Telekom on February 2008 [18]). The *export misconfiguration* happens when a BGP speaking router exports a route that it should filter.

Additionally, an *exposing attack* refers to the fact that an attacker can retrieve sensitive, private information that ASes do not want to share such as peering relations of a private network. This attack becomes severe with the involvement of the publicly available IRR system that many ISPs use to store their policies and validate others. In a *contamination attack* an attacker can forge the data inside a database system like IRR that hosts valuable information about ASes.

It is important to note that, in recent years, prefix hijacking (misconfiguration) and AS path spoofing (modification attack) are the most severe and usually deployed attacks [19].

Security solutions

Many security systems were designed as an answer to the attacks mentioned above. In this subsection we are going to describe the most important ones. The security solutions can be categorized into [16]:

- Cryptographic methods
- Non-cryptographic methods

From the cryptographic methods, worth mentioning representatives are: *Secure-BGP (S-BGP)* [20], *Secure origin BGP (soBGP)* [21], *Pretty Secure BGP (psBGP)* [22], *Secure Path Vector (SPV)* [23] [24], but the most important ones that we are going to describe as well, are *RPKI* and *BGPsec*. For non-cryptographic methods we have the *Inter-domain Route Validation (IRV)* system [25], the *IRR* database system and, as the most important one, the *BGP route filtering*.

The **Resource Public Key Infrastructure (RPKI)** cryptographic technology is one out of two, newest projects suggested to fulfill the Secure Inter-Domain Routing (SIDR) infrastructure requirements [26]. RPKI was standardized in RFC 6480 [27] and its goal was to satisfy the need for origin validation. Specifically, RPKI consists of a centralized database containing all the resource related information of the ASes with cryptographic protection. It is based on Resource Certificates (RCs) and Route Origin Authorizations (ROAs) so that valid statements can be created to express that an AS is allowed to announce specific IP prefixes.

BGP secure (BGPsec) is the second project suggested to fulfill SIDR. BGPsec, draft [28], tries to satisfy the path validation requirements. Its goal is the elimination of fake/wrong BGP paths that would attract traffic for a given destination. BGPsec is still a work in progress [29] [30].

Most of the proposed BGP security mechanisms are not in use. RIPE NCC administration has put a lot of effort the last 6 years into the development and promotion of RPKI system, which is considered to be along with BGPsec, the most complete security infrastructure for BGP. RPKI's adoption percentage was only 4% in 2014 according to [31] and BGPsec is under development and not officially standardized.

The reason that these mechanisms have trouble to be adopted or have not been adopted at all is that most of them (excluding RPKI) require modifications to the BGP messages structure. Particularly for the cryptographic methods, the computational cost increases due to heavy cryptographic operations that have to be supported and another important factor is that most of them do not offer backwards compatibility [16].

In contrast, from the non-cryptographic methods, **BGP route filtering** is the most important one as it constitutes the most effective and widely deployed technique for protecting against BGP vulnerabilities nowadays. One of its main usages is to apply the desired business relationships among ASes. It can also be used to establish access control lists on BGP routers stating which prefixes are allowed to be sent or received when exchanging BGP UPDATE messages [17] [32] [33].

2.2 Internet Routing Registry

IRR is the place where network operators store their AS routing policy. IRR constitutes one of the most important parts that helps BGP on the operation level and deployment. It is a distributed set of databases or repositories, the terms are going to be used interchangeably, containing the routing policies of many ASes [34] [35]. Currently, there are 26 both public and private routing registries [36], maintained on a voluntary basis, that network operators can query in order to search peering agreements, study optimal routes and (possibly automatically) configure their routers.

IRR has a hierarchical structure with Internet Assigned Numbers Authority (IANA) being at the root of the hierarchy. Below IANA are *five* Regional Internet Registries (RIRs), each of which is responsible for the management of IP address space and Autonomous System Number (ASN) allocation within a specific geographic region. For example, in Europe, RIPE NCC [37] is the responsible registry. The third part of the hierarchy varies. In some regions, we have the so-called National Internet Registries (NIRs), while in other regions we have the Local Internet Registries (LIRs) that describe the routing policies of customers of a specific ISP. Terms LIR and ISP are going to be used interchangeably along this report because in most regions they refer to

the same entity. All these databases taken together, they form the IRR.

The initial purpose of the IRR was the stability, security and consistency of the Internet, but nowadays its purpose is seriously under a lot of dispute. There are people who think its information is outdated, misleading and useless, while on the other hand, a lot of people believe that its information is valuable and its purpose needs to be preserved and encouraged by the network operators [13] [38].

2.2.1 IRR security considerations

According to [16] [25] [19] [17], the IRR functionality and structure raise security questions that we cannot ignore. First, there are some sensitive and confidential policy information such as peering agreements, concerning for example private networks, that ASes do not want to share with everybody. Due to these privacy concerns, many ASes choose to publish only part of their policy information. They do not care about keeping this information up to date and well-maintained. This, in turn, leads to possible misconfigurations as many ASes use IRR data to build their route filters.

Additionally, RIRs constituting the IRR database system are operating independently of each other under their own administration and at best they mirror each other (with mirrors being not always up to date), leading essentially to inconsistencies. IRR information often is inaccurate, meaning that accuracy of policy information can be valid by the time of its submission, but false when network operators request that information. Also, queries may return incomplete results (e.g. due to human errors) or a network operator can even get different results from queries to different routing registries.

Another problematic characteristic of today's IRR is that it provides no proper authorization/authentication of policy information changes to the registry. Also, the integrity of this information is questionable since everybody has access, including possible malicious users that may forge it.

These security concerns of the IRR database system, make more and more ASes' administrators less motivated in keeping their own policy records inside IRR updated, resulting in reducing its usefulness and harming its purpose. It is worth mentioning that RFC 2725 [39] is a proposal that tries to mitigate most of these concerns by introducing an authentication and authorization model so that integrity of stored information can be satisfied. However, this proposal needs to be re-examined for its applicability [17].

2.3 Routing Policy Specification Language

Network operators use BGP policies to enable their AS to communicate with other ASes and IRR to store these routing policies. Routing Policy Specification Language (RPSL) is another part that helps in BGP operation and constitutes the means to specify a routing policy in the IRR. It is

standardized in RFC 2622 [40] and it has been extended in RFC 4012 [41] in order to support IPv6 policies and multicast addressing (RPSLng).

RPSL is a vendor-neutral, object-oriented language that defines 13 classes of objects and its design serves a threefold purpose. First, the routing policy of an AS can be published in the IRR in an easy-readable and understandable format. Second, it provides high level classes that help in specifying a policy in a more comfortable and solid way [13]. Third, it can be converted into BGP routing configuration files.

The different classes of objects can describe a portion of the policy or who is the administrator of this policy. The main classes associated with the description of a policy are [13] [38]: `route class`, `route-set class`, `as-set class` and `aut-num class`.

Last but not least, we can use command-line tools like the WHOIS service which is based on the WHOIS protocol specification [42] in order to query a RIR database and get the routing policy of an AS. Additionally, there are tools that offer much more functionality than simply deploying queries, such as IRRToolSet [43], RPSLtool [44], BGPq3 [45], Netconfigs [46], etc.

2.3.1 RPSL adoption

Due to the fact that RPSL is very flexible by design, routing policies' specification can be very complicated. The level of accuracy of the descriptions largely varies as there can be many different ways to specify the same policy [13] [38].

In addition, there are many RPSL objects in the IRR that are not used but still there are references to them. Although RPSL is well-defined and organized in its core, it adds an extra high-level configuration step that most of ASes' administrators do not want as they see it as one more burden to their already demanding role. Due to this extra work, administrators are not motivated enough to learn how the RPSL concept actually works. As a result the amount of inaccurate data inside IRR increases due to missing updates.

Given these points, we can conclude that RPSL has not been an easy to adopt technology by network operators. It is not widely used and it proved to be difficult in practice as there are many operators that do not keep their RPSL policy records properly updated. In addition, there are also many operators that describe their policy by only using the basics of RPSL, avoiding to dive into more specifics to express their policy completely as it should be, because they lack the incentives to do so.

2.4 System design models

We can distinguish three system design models: the *centralized*, the *decentralized* and the *hybrid* system model. The final choice between these three patterns depends entirely on the requirements and needs of the system that is going to be designed [4].

Centralized model

According to this model, information is stored in a centralized database system and can be obtained by everyone. This type of system is being used in today's BGP routing procedure. ASes upload their policy information to the RIR responsible for their geographical location and then each RIR, according to its requests, distributes these policies to other ASes. Advantages of such a system is the ease of accessing the information, the easier communication between the domains and the enhanced security since there is only one location where policies are stored. However, among its drawbacks this approach defines a single point of failure and it accumulates traffic load to a central point.

Decentralized model

This structure implies that each domain is responsible for the maintenance and distribution of its own policy information. Some of its advantages are the reduced load, as this gets distributed among more than one domain and the supplied information is always accurate and up to date. Additionally, a decentralized approach is resilient as it has no single point of failure and can improve performance by distributing the query processing. Also, it can be scalable as we can easily add new components, without disturbing existing architecture. Some of the downsides are the increased complexity to the communication between domains and the difficulty of debugging.

Hybrid model

As its name states, this type combines characteristics of both the centralized and decentralized approaches. One of its main advantages is that it can be very effective. It can be designed in such a way that strengths of constituent approaches are maximized while the weaknesses are neutralized. Another advantage is that it is scalable. We can expand the system, in the same way as we can in a decentralized system, by easily adding new components, without disturbing existing architecture. Using a hybrid model system, the information provided is always accurate and up to date as it is when a decentralized model is used. Due to its centralized nature as well, the accessing of information becomes easier and the security can be enhanced. The combination of the approaches comes also with a price, which is the increased complexity of the design.

CHAPTER 3

METHODOLOGY

3.1 Literature study

The first and basic method that has been followed in order to cover the subject of this project was the literature study. The scope of this project is more theoretical and does not include any practical or experimental part. The sources that helped us during this study were mainly articles and RFCs, and of course Internet resources in general.

As Chapter 2 indicates, we started our research by studying the Internet technologies that this project is based on. These include the BGP protocol, the IRR distributed database system and the RPSL language. Combining these pieces together we were able to understand how BGP routing works in today's Internet as well as how the surrounding technologies, IRR and RPSL, help in the exchange of policy information between the administrative domains.

Subsequently our study was focused in BGP security threats and existing solution proposals. We also identified some IRR infrastructure considerations and difficulties that derived from the RPSL usage. This helped us understand the current BGP vulnerabilities and identify the weak parts of the existing BGP policy exchange system supported by IRR and RPSL. Lastly, literature study played a major role in establishing our system requirements.

3.2 Interviews and questionnaire

Apart from the theoretical knowledge obtained during the literature survey, we focused on gaining a practical background as well. In this part, two valuable sources of information have to be mentioned. First, the meetings with network operators who are experienced in BGP configuration. Second, a brief questionnaire got distributed among network operators in order to get an indication on what they usually do when a change to their policy information is required.

The individual meetings with the supervisors helped in gathering more general practical data about policy exchange procedure of different sizes of ISPs. Interviews with a few network operators (mostly of small ISPs) were very productive too, as they described the whole BGP policy of their network during the setup, update and maintenance phases. They pointed out that they do not use an automatic way as everything is done manually. All updates to the BGP configuration is done by hand and updates to policies are communicated through emails and face-to-face meetings. Digital authorization and authentication are not in place too. They pointed out the difficulties in using RPSL as well.

As a continuation to our effort to gain as much practical knowledge as possible, our next step was to create a brief questionnaire (2 questions). The content of the questionnaire was about the network operators' BGP policy update procedure. We signed up in *19* network operators' mailing lists from around the world to distribute the questionnaire to as many people as possible. The mailing lists that we signed up are presented in Appendix A.1. The statistical sample was *55 responses* and network operators could choose more than one answer to every question.

Figure 3.1 presents the means that network operators use in case that an update to their routing policy takes place. The big winner here is the "Immediate update of RPSL in RIR" supported by 38 out of 55 network operators, while 25 out of 55 responded that they communicate via email. Also, 4 out of 55 responded "Other" and only one answered "Communication via phone or fax". "Other" answers include "Update via a public community page" (2 times), "Minor changes are not communicated to peers and customers will be informed in case of an expected impact" and update does not take place at all.

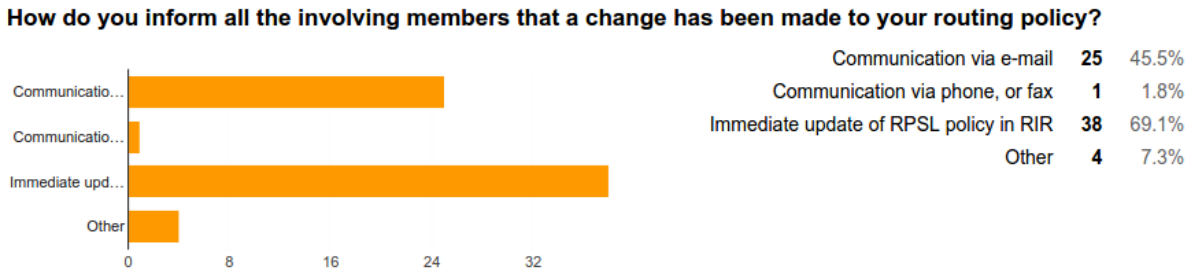

Figure 3.1: Question #1

At the second and last question, Figure 3.2, we tried to get an indication on what is the time interval until the actual update of RPSL when a policy changes. More than half of them, 29 out of 55 claimed immediately, while in the second place were the network operators who answered that RPSL's update is unrelated to the time that a policy changes (17 out of 55), which is the complete opposite from the winner option. A total of 13 out of 55 network operators responded "At the start/end of the week" or "Other", whereas only one replied "At the start/end of the month". The network operators (7 out of 55) that chose the "Other" option claimed that the time between a policy change and the actual RPSL update inside RIR is performed on daily basis (4 times),

every few months, every 48 hours or only after a customer request.

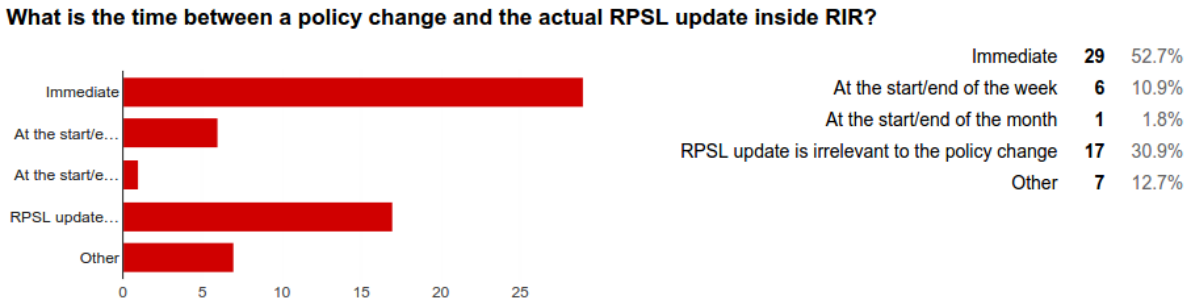

Figure 3.2: Question #2

Since this research involves a massive number of interconnected ASes, our statistical sample was small (only 55 responses) and we could not perform a statistical analysis in any case. However, the answers helped in defining our system requirements.

In summary, responses from both Questions #1 and #2 give us a indication that there is a need for an automatic way to exchange policies for both communication and update reasons. Network operators waste a lot of time in order to inform all involved members when a policy change happens, or they just updating their RPSL records in IRR and expecting the other parties to receive the new data as soon as possible. In either case, this is a time consuming procedure and requires also the extra high-level configuration step of RPSL, which has its practical difficulties that administrators are not willing to embrace. Additionally, results of Question #2 slightly indicate that RPSL is indeed difficult to adopt and use (17 out of 25 answered that RPSL update is unrelated to the policy change).

CHAPTER 4

SYSTEM DESIGN

4.1 Requirements

Based on the methodology described previously in Sections 3.1 and 3.2, derived the following requirements for our proposed system:

Table 4.1: System requirements

#	Requirements
1	Decentralization of policy information
2	Mapping between domains - policy service locations
3	Vendor-neutrality of routing policy language
4	Security (authorization/authentication)
5	Support for Policy Views (privacy)

Decentralization of policy information Every domain will be in charge of controlling and sharing its policy information. RFC 2769 [47] verifies there have been already some thoughts on designing a distributed approach. Decentralization can help us eliminate any inconsistencies. Policy information is going to be stored only on the originator's side and not to (maybe) multiple registries as can be the case with IRR. In addition, a better accuracy and completeness of policy information can be achieved as the originator will be the only one responsible for sharing its policy information. The originator will be also motivated enough to keep the policy up to date.

A decentralization approach is also encouraged by the answers to Questions #1 and #2 of the questionnaire (see Figures 3.1 and 3.2). RPSL is immediately updated whenever there are modifications to the routing policies and is used to communicate these changes to the interesting parties. This indicates that there is a need for an automatic way to exchange policies for both communication and update reasons.

Mapping between domains - policy service locations The Internet consists of a massive number of interconnected domains [1] and many of them share their routing policy information using the IRR system. In our decentralized approach, domains will interact with each other's local policy service locations for the exchange of policies. So, we need a way to manage the interconnected networks' policy information. Following this reasoning, a mapping entity between domains and policy service locations is necessary that will receive the initial requests and will be visible by every domain. This is why we need a centralized, redundant part for our system.

As reported from our literature study in Section 2.2 and also during our interviews, the IRR distributed database system already has most of the required functionality. The five existing RIRs are responsible for the assignment of ASNs and the allocation of IP address space, so they already maintain information of the existing ASes and they could host our central component. The functionality of the IRR concerning the storage and distribution of the policy information, as it is now, will not be taken into account.

Vendor-neutrality of routing policy language The only routing specification language that is being used right now is RPSL. In Section 2.3.1 we mentioned all the difficulties that were introduced during its adoption and usage. RPSL difficulties were also pointed out during the interviews and a little bit from Question #2 of the questionnaire (see Figure 3.2) in which 17 out 55 network operators answered that a policy update is unrelated to the actual RPSL record update. In this perspective, our proposed system has to be independent of any kind of routing policy specification language. We need our policy exchange system to be able to adapt to any routing policy description language.

Security (authorization/authentication) We already stated in Sections 2.1.2 and 2.2.1 that both BGP and IRR suffer from security vulnerabilities that cannot be ignored in any case. Although, our system is not a security-focused project, it aims to contribute to BGP security through the enhancement of the correctness and effectiveness of BGP filters. Following this reasoning, the communication between domains in our proposed system must be secure. Policy information exchanged must not be manipulated, falsified or read by third parties, and that is why we set as requirement for our system the mutual authentication of the participating domains.

Support for Policy Views (privacy) Policy views are the most fundamental part of our system and at the same time the innovation introduced by our system. By providing different policy views for different requesters we aim to preserve the confidentiality and sensitiveness of the policy data that most network operators desire. This sensitive policy data may include peering relationships and private agreements that should not be shared with everybody [16]. A certain policy view will be provided to every requester according to its domain identifier and its established business relation with the supplier. In any other case, a default policy view will be returned. By addressing the privacy concerns, we believe that operators will be also more

[1] For brevity, the *domain* term will be used when we refer to an *administrative domain or autonomous system (AS)*.

motivated to keep their policy information always up to date.

4.2 Decision making

This section is devoted to the description of our system model in conjunction with the security decisions that were made.

4.2.1 System model

The inspiration for our system model choice derives from [4]. Taking into account the system models that were discussed in Section 2.4 and the requirements of our system, in Section 4.1, we chose the *hybrid* model. All policy information is controlled and shared in a decentralized way by every domain independently, while a central entity is responsible for pointing to policy service locations.

Specifically, the centralized part of the system model is an entity with a supportive role, while the distributed part consists of two entities. A provider which shares/updates policy information and a requester which asks for policy information.

In our system model we can distinguish three main policy exchange components:

- **Policy Mapper (PM)** [1] **- centralized part** a repository responsible for the mapping between domains and policy service locations. Policy Mapper is discussed in more depth in Subsection 5.1.1.
- **Policy Provider (PP)** [1] **- decentralized part** stores, shares and updates policy information. Also, it allows other domains to register in order to send them notifications in case of a policy information update. Policy Provider is discussed in detail in Subsection 5.1.2.
- **Policy Requester (PR)** [1] **- decentralized part** asks for policy information. Policy Requester is discussed further in Subsection 5.1.3.

4.2.2 Security aspects

The security considerations that arise from the interaction between the components can be divided into two parts. The first part includes the communication of a PP or a PR with the PM and the second part includes the interaction between a PP and a PR.

Whenever a PR or a PP queries the PM to retrieve a domain record it is interested in, there is no need for authentication. The PM (which can be hosted by a publicly available RIR), maintains data which is only a pointer to a service and authentication will happen there.

[1] From now on, we are going to refer to each of the components using their unique abbreviations (PM, PP, PR).

The other part of security concerns includes the interaction between a PP and a PR and requires mutual authentication along with authorization (for policy views). A Public Key Infrastructure (PKI) is supported by our system for exchanging public keys through self-signed certificates in association with Transport Layer Security (TLS) communication. In order to secure this interaction, the following aspects have to be met:

1. The PM acts as a Trusted Third Party (TTP) and needs to be accessible by both PPs and PRs.

2. Each domain needs a public/private key pair to create a self-signed certificate and share it with the PM. The PR and PP components of the same domain use the same key pair to avoid maintaining a second registry.

3. PPs and PRs use their self-signed certificates over TLS to ensure that they talk to the right domain.

The PM could also act as a Certificate Authority (CA), but assuming that it will be hosted by a RIR, as we will see in Section 5.2, it would be wise to consider what the impact is going to be if RIRs get this additional role.

Last but not least, we should mention that we could also use raw public keys in TLS communication [48]. In general, self-signed certificates offer no security benefit over raw public keys. Self-signed certificates can offer a usability benefit, though, that allows the use of certificate-based software. Apart from that, there is no extra value of using self-signed certificates instead of raw public keys. We simply chose to maintain the *"certificate"* term because later in Subsection 6.2.1 we are going to discuss some security implementation ideas that involve explicitly the certificate concept.

CHAPTER 5

SYSTEM ARCHITECTURE

5.1 Architecture

The PM is the centralized part and can be hosted by RIRs. Every domain can act as a PP **and/or** as a PR. Both PPs and PRs take security into account and communication between them is by default signed and encrypted. Our system supports a PKI for the exchange of public keys through self-signed certificates. Since PMs can be hosted by RIRs, we can secure the creation/update of a policy service location off channel as authentication/authorization is already there for the RIRs.

5.1.1 Policy Mapper (PM)

A PM acts as a public map between domains and policy service locations. It can be accessed and consulted by every domain around the world, since it can be hosted by a RIR. It is responsible only for pointing to the policy service locations. No actual policy data is stored in it. Additionally, a PM holds the self-signed certificates of the domains that are registered in a RIR and that is why it is considered to be a TTP. It maintains one entry per domain, as shown in Figure 5.1. Its functionality is summarized below:

- Replies to queries from both PPs and PRs, supplying policy information details for a specific domain.
- Updates the data in the map registry for a given domain.

The entry fields for a domain are described below:

- **Domain ID** A domain identifier (e.g. ASN).

- **Policy Service Location** A pointer to the policy service for a given domain. This location will be a combination of `DOMAIN_NAME:PORT` [3]. It is a service responsible for the registration of other domains and the storage/share of the policy information in case of a domain acting as a PP. When a domain acts as a PR, this service is used for the notification updates. An arbitrary example could be `psl.asn1.org:port1`.
- **Certificate** A self-signed certificate that contains the public key and the digital signature for a given domain. Used for identity verification and encryption of data.

Figure 5.1: Policy Mapper registry entry

A PP can share its policy information, only after the registration of the domain to the PM. Every domain's administrator communicates with the organization that controls the PM in order to register the policy information details. The PM can be hosted by a RIR in our case, so RIR can authenticates/authorizes the domain through its login procedure. We will discuss the registration phase more in depth in Subsection 5.2.1.

5.1.2 Policy Provider (PP)

A PP holds the policy information and it is responsible for sharing according to requests. The functionality of a PP is summarized below:

- Distributes policy views of the main policy information to the PRs according to their domain intentifier. Requires mutual authentication. This architecture operation is described in more depth in Subsection 5.2.2.
- Registers domains that are interested in its policy information and would like to receive immediate update notifications when this changes. More details about this operation in Subsection 5.2.3.
- Sends notification updates to its registered domains in case of a policy information change. We analyze further this architecture operation in Subsection 5.2.4.

Policy Views are discrete pieces of the main policy information that describe how a network is operated (see Subsection 2.1.1). A PP distributes a policy view to a PR according to its unique domain identifier and their established business relationship and agreement. If no business relationship and agreement is established, it distributes a *default* policy view. The following flowchart in Figure 5.2 depicts this decision in a PP:

[3] a Uniform Resource Identifier (URI)

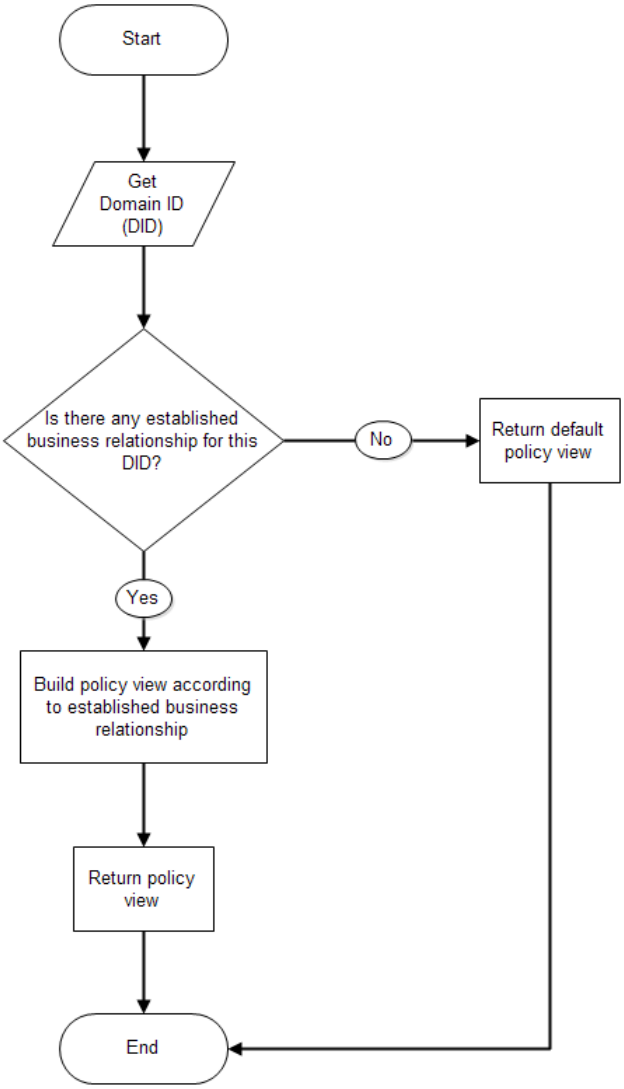

Figure 5.2: Policy view handling

During the (first) interaction between a PP and a PR, the latter can choose to register to the former by supplying its domain intentifier (e.g. ASN). In this way, in case of a policy information update the PP will notify the domains registered to its database immediately. The PP is going to consult the PM to retrieve the required policy service locations.

5.1.3 Policy Requester (PR)

A PR asks for policy information. Its functionality includes:

- Asks the PM to obtain the domain entries it is interested in and then it communicates directly with the PPs using the policy service locations. Requires mutual authentication for the direct communication with the PP, but not for the communication with the PM.

- Receives policy update notifications from PPs that previously had been registered to. Then the administrator of the domain decides when the update will take place.

5.2 Architecture operations

In this section, a description of the interaction between the three components is presented. We describe each operation separately and we assume that the centralized part of our system, PM, is hosted by a RIR. The main functionality of the system can be summarized as follows:

First, the administrator of a domain registers a domain's policy information details inside the PM, after he/she logged in to RIR's portal. Then, a PR asks the PM for an entry of a domain. The PM returns the requested entry and then the PR uses the policy service location to send a signed and encypted request (using the public key of the PP) to a PP, in order to pull the policy view for its domain. At this point, mutual authentication takes place over TLS using the certificates that both domains have shared previously with the PM. If authentication is successful, the PP returns a policy view to the PR. PRs can also register themselves to PPs, in order to receive an immediate update notification in case of a policy update. In case that a PP changes the policy service location, it is an administrator's responsibility to update the domain record inside PM registry.

5.2.1 Registration of a domain to the Policy Mapper

Figure 5.3 illustrates the interaction between the administrator of a domain and the PM (hosted by a RIR) during the registration of its policy information details.

First, we assume that by the time a network operator signs up to a RIR database, a new entry is created inside the PM with the network's unique domain identifier (e.g. ASN). Second, we assume that the communication between the domain and the PM is secure since the network operator of the domain presents his/her RIR portal credentials during login.

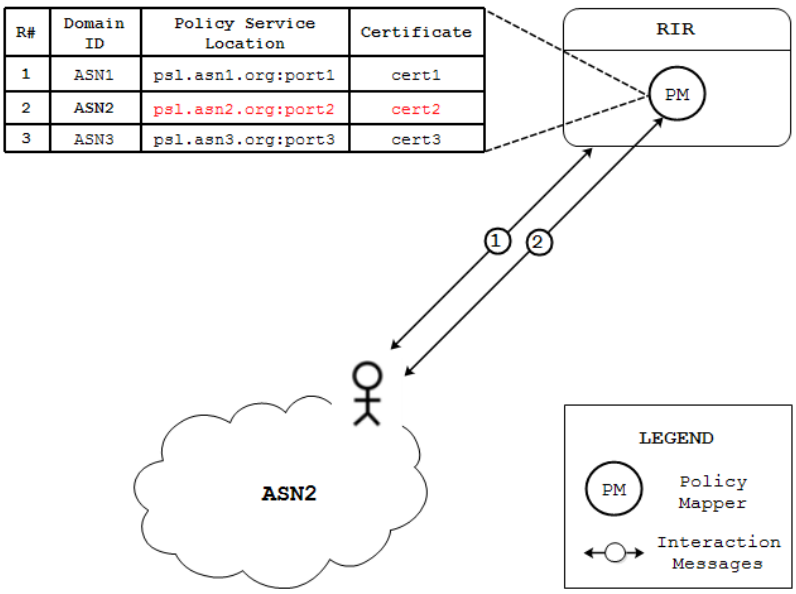

Figure 5.3: Registration phase of ASN2 domain's policy information details

Specifically, the interaction messages that take place between the administrator of ASN2 domain and the PM have the following meaning:

1. The administrator of the ASN2 domain interacts with the RIR portal, by presenting his/her credentials (username and password) in order to login. If authentication is successful, the administrator navigates to the PM dashboard.

2. The administrator of the ASN2 domain interacts with the PM, by editing the policy service location and uploading the self-signed certificate of the underlying domain. The administrator saves the changes and the PM updates the policy information details for that domain.

Unregistration phase A domain wants to free its allocated ASN and IP address space. The administrator of the domain logs into the RIR portal, navigates to the PM dashboard, deletes the policy service location and the corresponding certificate and saves the changes. We could rely on RIR for the deletion of policy information details, but in this way the domain preserves the control of its security as far as de-registration concerns. Then, the domain administrator can send an email with a de-registration request to the RIR administrators. Subsequently, RIR administrators delete the underlying domain ID entry and send a confirmation email to the domain administrator. This is a recommended or possible scenario.

5.2.2 Policy retrieval

The main functionality of a PP is the distribution of policy views to PRs. It supplies a policy view according to their unique domain identifier and their established business relationship and agreement. In any other case, it provides a default policy view. Figure 5.4 illustrates two PRs that attempt to retrieve policy information from a PP.

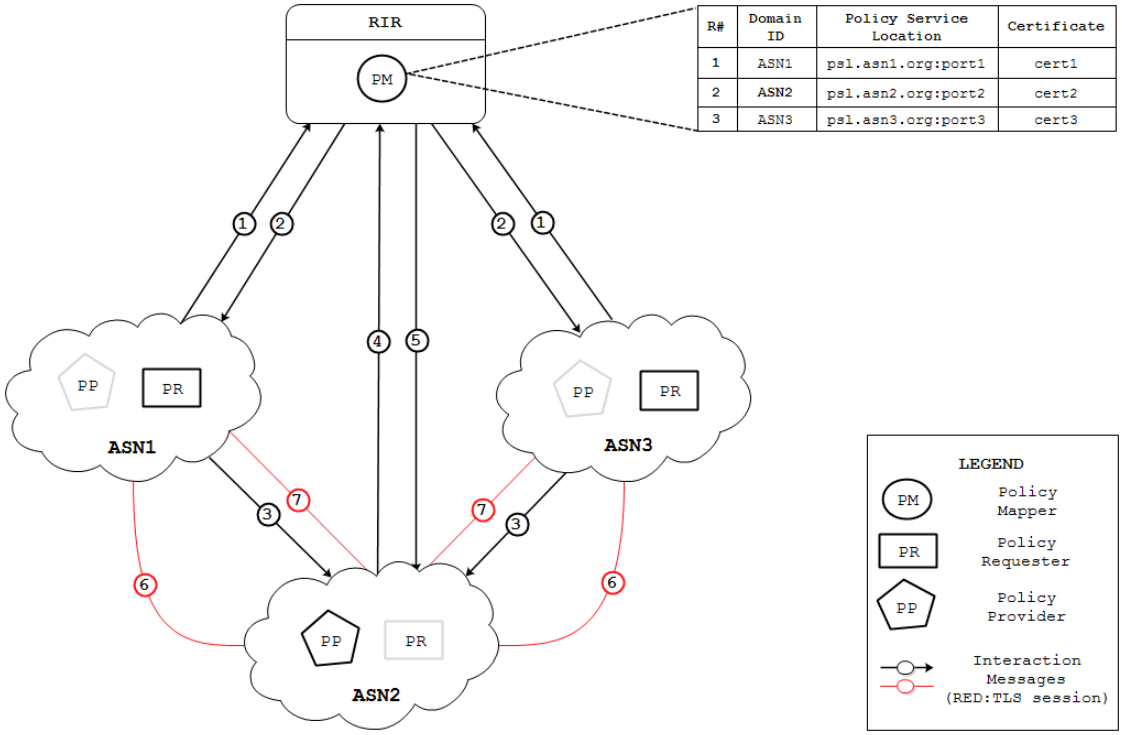

Figure 5.4: Policy retrieval of ASN1 and ASN3 Policy Requesters from ASN2 Policy Provider

The interaction messages exchanged are summarized as follows:

1. ASN1 and ASN3 PRs ask the PM for the registry entry of ASN2 domain.
2. PM returns the requested entry to both PRs, which is record #2.
3. ASN1 and ASN3 PRs send a signed and encrypted policy request to the policy service location of ASN2 PP using the public key of ASN2 PP.
4. ASN2 PP, receiving the requests, decrypts them using its private key and consults the PM to retrieve the self-signed certificates of ASN1 and ASN3 domains.
5. PM returns the self-signed certificates of ASN1 and ASN3 domains to ASN2 PP.
6. ASN2 PP verifies the received signed requests. Only then, a TLS session is initialized and mutual authentication takes place between the ASN2 PP and ASN1, ASN3 PRs. Starting

TLS session after the verification protects against potential DoS attacks by unauthenticated requests.

7. ASN2 PP distributes a policy view to each of the PRs.

5.2.3 Registration of a Policy Requester to a Policy Provider

A PP provides the ability to a PR to register to it, in case that it wants to receive an immediate notification as soon as a policy update happens. Registration is recommended in case that there is an established business relationship and agreement between the two entities. Figure 5.5 illustrates a PR that attempts to register to a PP.

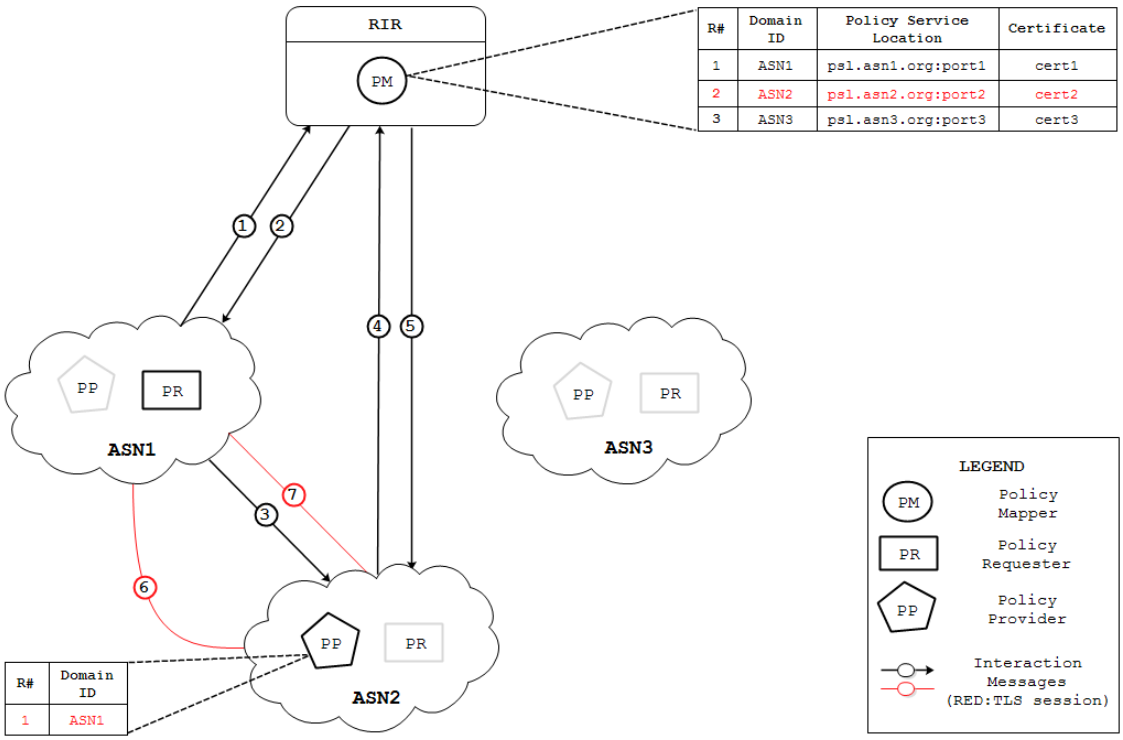

Figure 5.5: Registration of ASN1 Policy Requester to ASN2 Policy Provider

The interaction messages exchanged have the following meaning:

1. ASN1 PR queries the PM to retrieve the registry entry of ASN2 domain.
2. PM returns the requested entry, which is record #2.
3. ASN1 PR sends a signed and encrypted registration request to the policy service location of ASN2 PP using the public key of the ASN2 PP.
4. ASN2 PP decrypts the request using its private key and consults the PM to retrieve the self-signed certificate of ASN1 domain.

5. PM returns the self-signed certificate of ASN1 domain to ASN2 PP.

6. ASN2 PP verifies the received signed request. Only then, a TLS session is initialized and mutual authentication takes place between the ASN2 PP and ASN1 PR.

7. ASN2 PP informs the ASN1 PR for the status of the registration (successful/failed).

Unregistration phase A PR wants to unregister itself from a PP due to break of business relationship (dismissal of contract). This is the same procedure as the registration one, but instead of a registration request, a PR sends a de-registration request. A PR sends its signed and encrypted de-registration request to the corresponding PP (consulting first the PM). The PP, decrypts the request, verifies the identity of the PR (consulting first the PM), deletes its domain ID entry from its registry and then sends a confirmation reply back to the PR.

5.2.4 Policy update and notification

As long as a PR registered itself to a PP, it will get immediate notifications when a policy update occurs. Figure 5.6 presents a scenario where a registered PR receives a notification, while another one does not (although it is active).

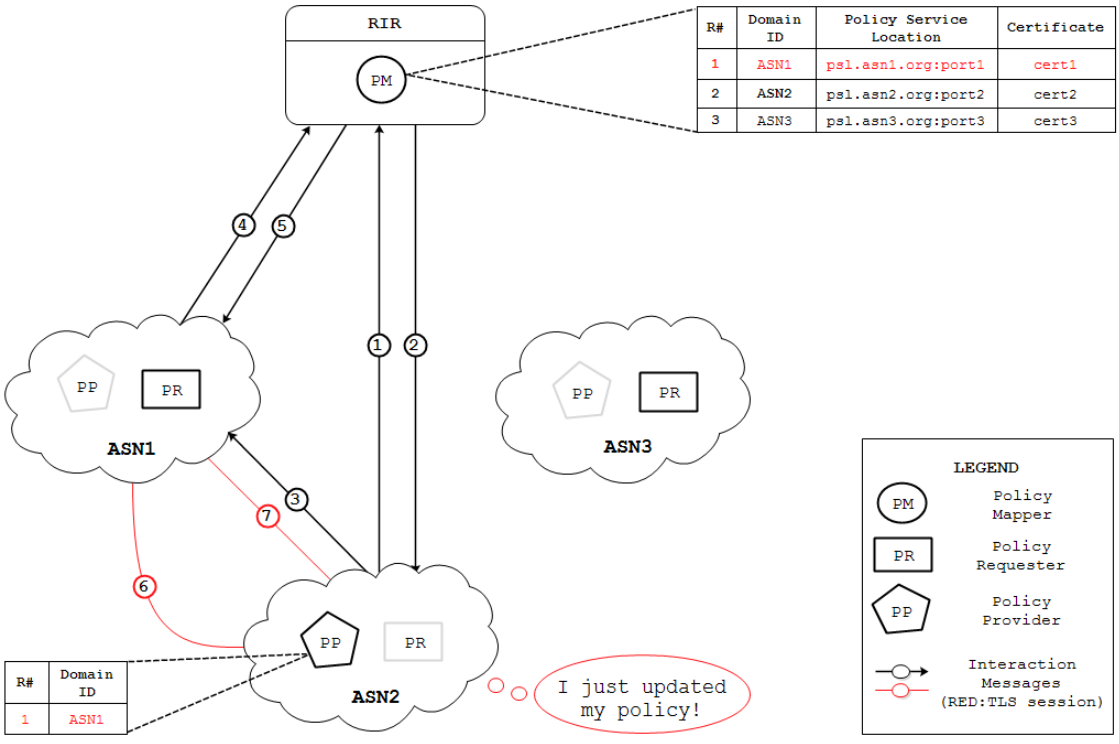

Figure 5.6: Update policy of ASN2 Policy Provider and update notification to registered to it ASN1 Policy Requester

Once more, a description of the interaction messages is supplied:

1. ASN2 PP just updated its policy information and checks its registry for registered PRs. Only ASN1 PR is registered, so next PP consults the PM to obtain ASN1 domain entry.
2. PM returns the requested entry, which is record #1.
3. ASN1 PP uses the policy service location to reach the ASN1 PR and sends a signed and encrypted policy update notification which includes its domain identifier, using the public key of ASN1 PR.
4. ASN1 PR, receiving the notification request, it consults the PM to retrieve the self-signed certificate of ASN2 domain.
5. PM returns the self-signed certificate of ASN2 domain to ASN1 PR.
6. ASN1 PR verifies the received signed policy update notification. Only then, a TLS session is initialized and mutual authentication takes place between the ASN2 PP and the ASN1 PR.
7. ASN1 PR informs the ASN2 PP for the status of the policy update notification (successful/-failed).

5.2.5 Certificate expiration

During the initialization of the system, it is an administrator's obligation to upload the certificate of the domain to the PM. When an expiration is about to happen, following the same reasoning, an administrator must renew the certificate immediately.

In every interaction between the components of our system, the self-signed certificates are obtained dynamically. In this way we can restrict authentication failovers. There are a lot of malicious uses that can lead to an identity verification or encrypted communication failure. One of these is when a certificate expires and does not get renewed immediately. Once a certificate expires, it becomes invalid and the domain that owns it cannot be verified from other domains or establish a secure connection. As a result, a domain can neither share its policy information nor request other domains' policy information.

We consider that the PM hosted by a RIR is acting as a TTP, thus being the only valid source of information. The RIR portal could prompt the administrators to renew the self-signed certificates of their domains prior to the expiration date. It is every domain administrator's responsibility to keep the PM record up to date.

CHAPTER 6

DISCUSSION

6.1 Scalability and high availability

Our system design is based on the hybrid model because this one was the most suitable to our needs. One of the main advantages of this model, as we discussed in Section 2.4, is its scalability. It is easy to increase the size of facilitated ASes by simply adding new components, without disturbing existing architecture. Furthermore, our system is a rather new idea and there is no other standardized system with related functionality that we could compare (in practice) to our proposal in terms of scalability and efficiency.

However, both PM and PP components are single points of failure in our system design. If the PM becomes unavailable there will be a severe degradation in performance. A domain will be unable to retrieve the policy service locations it is interested in and establishment of secure communications will be impossible as well. Additionally, if a PP fails then a domain will be unable to share and update its policy information. The hybrid architecture is very flexible and there are ways to mitigate these. We can ensure that failure of these two single components would not cause the entire system to fail by using redundancy. Examples of this strategy include deploying multiple, replicated instances of both PP and PM components. Although, replication as part of a redundant solution will provide high availability, we should mention that it will cause additional overhead in terms of module management.

6.2 Implementation ideas

There are three technologies that could be used to support the required security in our system, namely Resource Public Key Infrastructure (RPKI), Hypertext Transfer Protocol Secure (HTTPS) and DNS-Based Authentication of Named Entities (DANE). Also, as far as the routing policy

description language concerns, the main candidate is the already in place RPSL. However, we will discuss about Routing Documentation Language (RDL) and YAML Ain't Markup Language as well.

6.2.1 Security candidates

RPKI would be the perfect match for our system security as it is a technology hosted by every RIR and it is being used for origin validation in BGP (see Sub-subsection 2.1.2 for more information). We could store the Resource Certificate (RC) of every domain to the certificate field of every PM entry. In this way, PR and PP domains would be able to use an already existing certificate which is hosted by the RIR. However, RPKI Resource Certificates arise two important concerns.

First, according to RFC 6480 [27]: *"[...] the subject names used in certificates are not intended to be "descriptive". That is, the resource PKI is intended to provide authorization, but not authentication. [...]"*, meaning that we cannot verify the identity of the communicating domains.

Second, the key pairs are generated using a Hardware Security Module (HSM) and private keys cannot be extracted from the HSM in unencrypted form. This means that unencrypted private keys are stored only in RIRs [49]. Automation of all cryptography complexity was one of RPKI's goals, but at least private keys should rest at the domains' side.

It is worth mentioning, though, an interesting idea introduced by Dragon Research Labs [50]. Software is offered to run your own CA that securely interfaces with the RIPE NCC parent system. This would be the perfect match as each domain would be in complete control of its RC and the corresponding private key. Also, each domain can choose to publish the certificate by itself or let another party do it on its behalf. Nonetheless, this software is currently under testing and not in production.

HTTPS protocol is the second choice that we could use for our system security. In HTTPS, a SSL web certificate needs to be purchased from a Certificate Authority (CA) such as Comodo or Verisign, which will be installed in a web server. It would be wise to assign the role of the CA to every PM (hosted by a RIR in our examples) because it holds the policy information details for every domain. However, as we discussed earlier in Subsection 4.2.2, we should consider what the impact is going to be if RIRs get this additional role.

Furthermore, the web certificate would be useful for the PRs to verify that they talk to the right domain. However, with HTTPS there is also the need for client certificates, because every PP should be able to authenticate the requesters that connect to it as well. According to RFC 5280 [51], by enabling the "Extended Key Usage" option we can state that the web certificate will be used as both server and client certificate, meaning that it will be used by both PRs and PPs.

The third and last security implementation idea involves the DNS-Based Authentication of Named Entities (DANE). DANE, specified in RFC 6698 [52], is a protocol that binds X.509 certificates, commonly used for TLS, with Domain Name System (DNS) names using Domain Name System Security Extensions (DNSSEC). It uses a so-called TLSA DNS resource record

(RR) in order to associate a TLS certificate or public key with the domain name where the record belongs. The most important field of a TLSA record is the `certificate usage` field. It provides the association that will be used to match the certificate presented in the TLS handshake and can have four values (0-3).

The design of our system, involves self-signed certificates. Every domain maintains its own and shares it with the PM, in order to be verified by others and establish secure connections over TLS. Using DANE the TLSA records' `certificate usage` field should have the value 1. This value indicates that these exact self-signed certificates should be matched with the self-signed certificates that each component (PP or PR) retrieves from the PM [53]. In this way, we can ensure that we talk to the right end entity as the TLSA record is going to be signed using DNSSEC. In addition, the PM is no longer storing certificates and it is no longer a TTP as trust is now put into DNSSEC. It is worth mentioning that the PM cannot be entirely excluded. Even if all domain administrators agreed upon a well-known port number and a sub-domain (may be through a standardization) about policy service location (e.g. `_1234._tcp.domain.org`), there is still a need for an entity responsible for the mapping between domain identifiers (e.g. ASNs) and domain names.

6.2.2 Policy specification languages

Our research led us to the conclusion that there is a need for a structure-based, human-readable language that provides a correlation between router configurations and policies. This correlation will help us to take the maximum advantage of the policy views our system provides.

RPSL is the only routing policy specification language that is being used nowadays. In Section 2.3.1 we mentioned the difficulties that arose with its usage during the description of policies. Its object-oriented nature makes it very flexible, producing in many cases very complex policy descriptions. Due to this flexibility, the level of accuracy of descriptions largely varies as well.

An alternative of RPSL is the RDL language. It is part of the Extendible Next Generation Routing Information Toolkit (ENGRIT) project that started off in 2014. ENGRIT is a toolset that is being developed from NLnet Labs in collaboration with a few industry partners who provide feedback on the design and implementation. The RDL specification language has been presented at the UKNOF 28, IEPG (at the IETF 89), and RIPE 68 meetings in 2014. Per Gregers Bilse implemented a compiler and RDL's goal was to be more expressive, easier to use and read, and correspond to the current practices in routing configuration [54].

RDL intended to reuse parts of RPSL, so it is an object-oriented language as well, but more importantly it intended to describe BGP topologies in a simple way, cover both iBGP and eBGP peering relationships and fully identify routing policies. It is based on three main components, namely *zones*, *routers* and *peers*. These objects have a number of attributes and the whole structure is similar to file system directories. Specifically, *zones* are containers for similar policies. Lastly, RDL has been designed in order to help programming an AS, not configuring the BGP

routers [55].

YAML's syntax could also be used instead of RPSL. YAML is a general purpose, human-readable data serialization language that is commonly used for configuration files. Most programs use one or more configuration files in YAML format. For example, YAML is already used to describe BGP neighbors configuration. The BGP peering relations and customers connected to a router can be listed in a YAML configuration file containing three YAML documents (lists of variables separated by "-"). Following the same reasoning, we could use YAML syntax to describe BGP policies. Additionally, it is worth mentioning that YAML is used by networking automation tools like Ansible and network operators are already accustomed with it.

CHAPTER 7

CONCLUSION

Network operators use a centralized system model (RIRs) in order to exchange their BGP policy information. During our literature survey, we identified that this system raises a lot of security concerns and difficulties that need to be addressed.

The architecture proposed in this report tries to support the policy exchange system through a different angle. Instead of a centralized approach, a hybrid one is presented. Policy information is exchanged/updated in a decentralized way, while a central component is responsible for the mapping between domains and policy service locations.

Despite of its complexity in design, this hybrid model promises flexibility and scalability to our architecture. Our system is defined in a straightforward way and the components are simple and well-defined. The system can be expanded by easily adding new components, without disturbing the existing architecture.

In addition, although our system is not a security-focused mechanism, it can contribute indirectly to BGP security. In our hybrid approach, ASes communicate directly with each other to exchange/update policies and RIRs simply point to their policy information for quick access. Our system also makes use of a Public Key Infrastructure (PKI) for providing authentication and authorization. Furthermore, update notifications are sent only to involved ASes, and each AS obtains only the part of a policy that concerns it. We believe this proposal will supply more incentives to network operators to keep their policy information up to date and accurate. Confidentiality and sensitiveness of their policy data will be preserved, and this in turn will enhance the effectiveness of BGP route filters.

CHAPTER 8

FUTURE WORK

There are four main directions for future work. First and most importantly, we need to build a proof-of-concept. We have established the requirements of our system and the research area in which it is defined. A prototype can be our guide to compare the theoretical research with the practical experimentation.

A second direction has as prerequisite an already implemented Proof of Concept and refers to the size of ISPs. Our design is mainly based on small (e.g. SURFnet with 25 ASes and 204 prefixes [1]) and medium size ISPs (e.g. Hibernia Networks B.V. with 1526 ASes and 11759 prefixes [1]). It would be interesting to test our prototype in a large scale scenario with big size ISPs (e.g. NTT America, Inc. with 18778 ASes and 161697 IPv4 prefixes [1]). Bigger ISPs maintain a lot of peering relations and their administrators need to update the policy information periodically. In this perspective, it would be thought-provoking to evaluate how our system would behave in such a scenario.

A third direction would be the correctness of policy information. In our system design there is no functionality that checks the correctness of policy information before it is used by a requesting domain. In particular, we could extend the functionality of the PR component for this cause. The PR could be designed to compare the received policy view with the policy information that the PP keeps locally. In case that these two do no match, the PR could inform the PP back. In this way, we could ensure that the administrator of the requesting domain will have the correct policy information (according to established business relation and agreement) to build the BGP filters for his/her domain in the most effective way.

A fourth and last direction is to conduct a research on routing policy specification language alternatives to supersede the existing RPSL. This language is the only one being used nowadays

[1] Statistics were obtained from the Center for Applied Internet Data Analysis (CAIDA) website [56].

and as we discussed in 2.3.1, its practical usage comes with a lot of difficulties. Two worth mentioning alternatives are RDL and YAML. Their basics were discussed in 6.2.2.

APPENDIX A

MAILING LISTS

The mailing lists that we signed up, in order to distribute the questionnaire to as many network operators as possible, can be seen in the following table:

Table A.1: Mailing lists

#	List Name	Description
1	denog	German Network Operators Group
2	ausnog	Australian Network Operators Group
3	enog	Eurasia Network Operators Group
4	panog	Pakistan Network Operators Group
5	bdnog	Bangladesh Network Operators Group
6	mynog	Malaysia Network Operators Group
7	sdnog	Sudan Network Operators Group
8	afnog	African Network Operators Group
9	sgnog	Singapore Network Operators Group
10	nznog	New Zealand Network Operators Group
11	itnog	Italian Operators Group
12	nlnog	Netherlands Network Operators Group
13	grnog	Greek Network Operators Group
14	btnog	Bhutan Network Operators Group
15	uknof	UK Network Operators Forum
16	berlin	DENOG Stammtisch Berlin
17	pdb-tech	Peering DB tech list
18	routing-wg	RIPE NCC Routing Working Group
19	db-wg	RIPE NCC Database Working Group

BIBLIOGRAPHY

[1] Wikipedia, "Internet Routing Registry — Wikipedia, the free encyclopedia." https://en.wikipedia.org/wiki/Internet_Routing_Registry.
Accessed online on 07-10-2016.

[2] McPherson, Danny, Blunk, Larry, Amante, Shane, Osterweil, and Eric, "IRR & Routing Policy Configuration Considerations," 2015.

[3] S. Vouteva and T. Turgut, "Automated configuration of BGP on edge routers," System and Network Engineering MSc, University of Amsterdam, 2015.

[4] R. Koning, M. Zivkovic, S. Konstantaras, P. Grosso, C. de Laat, and F. Iqbal, "Architecture for Exchanging Topology Information in Multi-domain Environments," University of Amsterdam and Delft University of Technology, 2015.

[5] Wikipedia, "Distributed hash table — Wikipedia, the free encyclopedia." https://en.wikipedia.org/wiki/Distributed_hash_table.
Accessed online on 19-10-2016.

[6] Wikipedia, "Freenet — Wikipedia, the free encyclopedia." https://en.wikipedia.org/wiki/Freenet.
Accessed online on 21-10-2016.

[7] Wikipedia, "Key-based routing — Wikipedia, the free encyclopedia." https://en.wikipedia.org/wiki/Key-based_routing.
Accessed online on 08-10-2016.

[8] K. Loogheed, Y. Rekhter, and T. Watson, "A border gateway protocol (BGP)," tech. rep., RFC, 1989.

[9] Rekhter, Yakov, Li, Tony, Hares, and Susan, "A border gateway protocol 4 (BGP-4)," tech. rep., 2005.

[10] Wikipedia, "Peering — Wikipedia, the free encyclopedia." https://en.wikipedia.org/wiki/Peering.
Accessed online on 29-10-2016.

[11] L. Vanbever, B. Quoitin, and O. Bonaventure, "A hierarchical model for BGP routing policies," in *Proceedings of the 2nd ACM SIGCOMM workshop on Programmable routers for extensible services of tomorrow*, pp. 61–66, ACM, 2009.

[12] M. Caesar and J. Rexford, "BGP routing policies in ISP networks," *IEEE network*, vol. 19, no. 6, pp. 5–11, 2005.

[13] G. Siganos and M. Faloutsos, "Analyzing BGP policies: Methodology and tool," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1640–1651, IEEE, 2004.

[14] RIPE NCC, "RIPE Routing Working Group Recommendations on Route Flap Damping." `https://www.ripe.net/publications/docs/ripe-580`, 2013.
Accessed online on 30-10-2016.

[15] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *ACM SIGCOMM Computer Communication Review*, vol. 32, pp. 3–16, ACM, 2002.

[16] E. Kim, K. Nahrstedt, L. Xiao, K. Park, *et al.*, "Identity-based registry for secure interdomain routing," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 321–331, ACM, 2006.

[17] L. Blunk and M. Karir, "Internet Hardening via Routing Registries," 2005.

[18] Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, "How YouTube was "Hijacked"," *Internet Society, Reston, VA. http://web.mit.edu/6.02/www/s2012/handouts/youtube-pt.pdf*, May 2009.

[19] M. Zhang, "On the State of the Inter-domain and Intra-domain Routing Security,"

[20] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected areas in Communications*, vol. 18, no. 4, pp. 582–592, 2000.

[21] R. White, "Securing BGP through secure origin BGP (soBGP)," *Business Communications Review*, vol. 33, no. 5, pp. 47–53, 2003.

[22] T. Wan, E. Kranakis, and P. C. van Oorschot, "Pretty Secure BGP, psBGP," in *NDSS*, 2005.

[23] Y. Hu, A. Perrig, and M. Sirbu, "SPV: Secure path vector routing for securing BGP," in *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 179–192, ACM, 2004.

[24] A. Narayanan, "A survey on BGP issues and solutions," *arXiv preprint arXiv:0907.4815*, 2009.

[25] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. D. McDaniel, and A. D. Rubin, "Working around BGP: An Incremental Approach to Improving Security and Accuracy in Interdomain Routing," in *NDSS*, 2003.

[26] J. Durand, I. Pepelnjak, and G. Doering, "BGP operations and security," tech. rep., 2015.

[27] M. Lepinski and S. Kent, "RFC 6480: an infrastructure to support secure Internet routing. Internet Engineering Task Force (IETF)," 2012.

[28] S. Bellovin, R. Bush, and D. Ward, "Security requirements for BGP path validation," tech. rep., 2014.

[29] M. Lepinski *et al.*, "An overview of BGPSEC," 2015.

[30] G. Huston and R. Bush, "Securing BGP with BGPsec," in *The Internet Protocol Forum*, vol. 14, 2011.

[31] A. Malhotra and S. Goldberg, "RPKI vs ROVER: comparing the risks of BGP security solutions," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 113–114, ACM, 2014.

[32] O. Nordström and C. Dovrolis, "Beware of BGP attacks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 1–8, 2004.

[33] K. R. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A Survey of BGP Security Issues and Solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2010.

[34] Wikipedia, "Internet Routing Registry — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/wiki/Internet_Routing_Registry`.
Accessed online on 30-10-2016.

[35] Merit Networks, "Overview of the IRR." `http://www.irr.net/docs/overview.html`.
Accessed online on 30-10-2016.

[36] Merit Networks, "List of Routing Registries." `http://www.irr.net/docs/list.html`.
Accessed online on 30-10-2016.

[37] RIPE NCC, "RIPE Database Documentation." `https://www.ripe.net/manage-ips-and-asns/db/support/documentation/ripe-database-documentation`.
Accessed online on 30-10-2016.

[38] G. D. Battista, T. Refice, and M. Rimondini, "How to extract BGP peering information from the internet routing registry," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pp. 317–322, ACM, 2006.

[39] C. Villamizar, C. Alaettinouglu, D. Meyer, and S. Murphy, “C. Orange," Routing Policy System Security,” tech. rep., RFC 2725, December, 1999.

[40] C. LAETTINOGLU, C. VILLAMIZAR, E. GERICH, *et al.*, “IETF RFC 2622,” *Routing policy specification language (RPSL)*, 1999.

[41] L. Blunk, J. Damas, F. Parent, and A. Robachevsky, “Routing policy specification language next generation (RPSLng),” tech. rep., 2005.

[42] L. Daigle, “WHOIS protocol specification,” tech. rep., RFC 3912, Internet Engineering Task Force (March 2004) 206 E. Passerini et al, 2004.

[43] Github, “IRRToolSt.” `http://www.netconfigs.com/`.
Accessed online on 31-10-2016.

[44] Github, “RPSLtool.” `https://github.com/rfc1036/rpsltool`.
Accessed online on 31-10-2016.

[45] Github, “BGPq3.” `https://github.com/snar/bgpq3`.
Accessed online on 31-10-2016.

[46] Netconfigs.com, “Netconfigs.” `http://www.netconfigs.com/`.
Accessed online on 31-10-2016.

[47] C. Villamizar and R. Govindan, “Routing Policy System Replication,” 2000.

[48] Wouters, P and Tschofenig, H and Gilmore, J and Weiler, S and Kivinen, T, “Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS),” tech. rep., 2014.

[49] RIPE NCC, “RIPE NCC RPKI (Resource Public Key Infrastructure) Certification Practice Statement (CPS).” `https://www.ripe.net/publications/docs/ripe-549`.
Accessed online on 05-11-2016.

[50] RIPE NCC, “Using the RPKI system.” `https://www.ripe.net/manage-ips-and-asns/resource-management/certification/using-the-rpki-system`.
Accessed online on 06-11-2016.

[51] P. Yee, “Updates to the internet X.509 public key infrastructure certificate and Certificate Revocation List (CRL) profile,” 2013.

[52] P. Hoffman and J. Schlyter, “The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA,” tech. rep., 2012.

[53] Internet Society, "The DANE Protocol - DNS-Based Authentication of Named Entities." http://www.internetsociety.org/deploy360/resources/dane/.
Accessed online on 18-11-2016.

[54] NLnet Labs, "AnnualReport 2014: For an Open Internet." https://www.nlnetlabs.nl/annualreports/annualreport2014.pdf.
Accessed online on 15-11-2016.

[55] P. G. Bilse and B. Overeinder, "A programmatic approach to generating router configurations." https://indico.uknof.org.uk/event/30/contribution/21/material/slides/0.pdf.
Accessed online on 15-11-2016.

[56] Center for Applied Internet Data Analysis (CAIDA), "AS Rank: AS Ranking." http://as-rank.caida.org/?mode0=as-ranking&n=50&ranksort=1.
Accessed online on 05-11-2016.