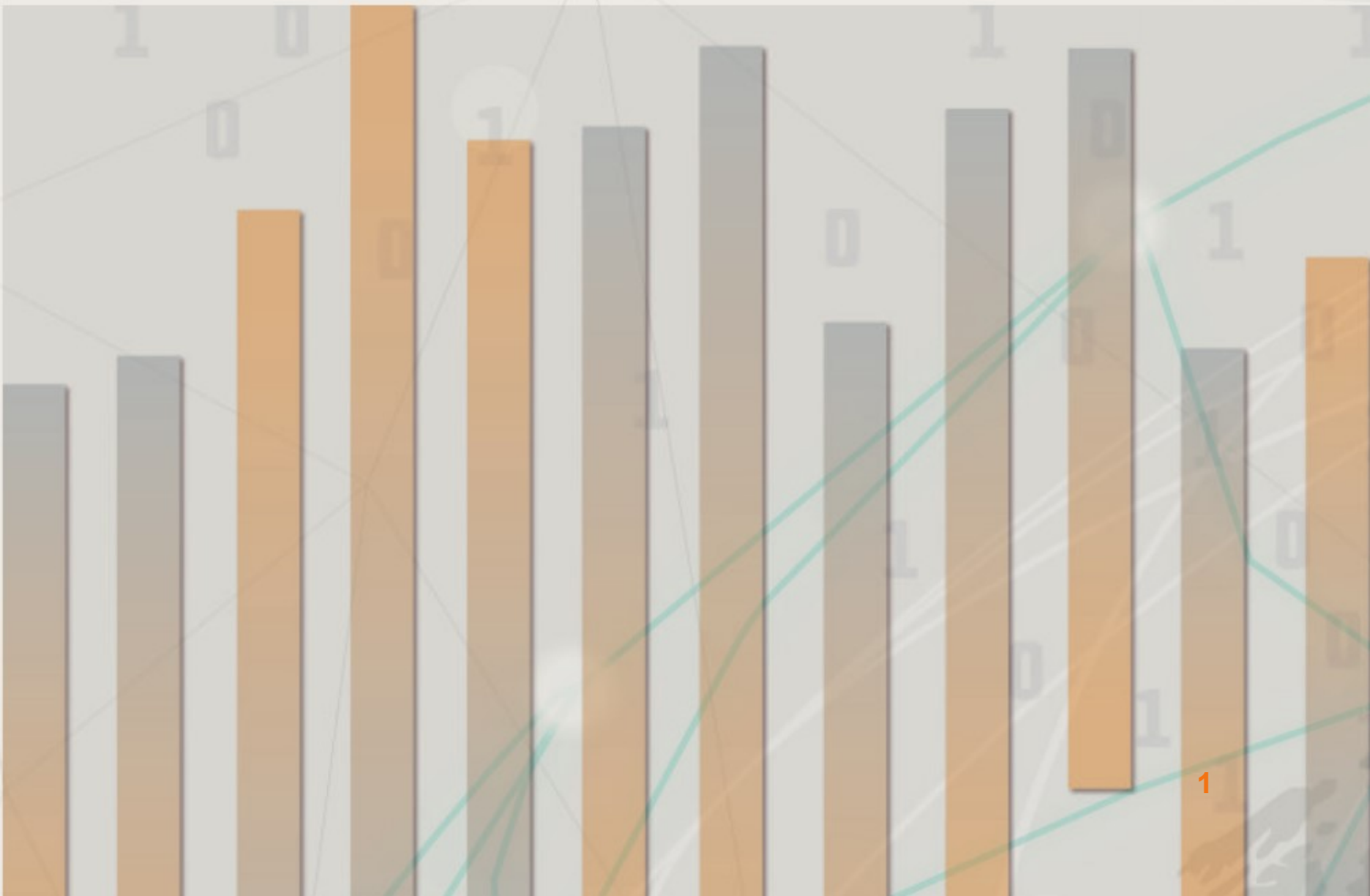
REISS ROMOLI

For a safer Internet

BGP RPKI: instructions for use

Flavio Luciani
Tiziano Tofoni



Published in 2020

Background vector: freepik.com

COPYRIGHT NOTES

1 – INTRODUCTION

Any discussion around aspects of Internet security cannot begin without illustrating the possible types of attacks and providing an analysis of vulnerabilities. It is clear that the entire Internet is vulnerable to attacks to its routing protocols. *Border Gateway Protocol* (or BGP), from this point of view is no exception, and being the most important of the Internet routing protocols, is where most attention should be focused, in terms of protection.

1.1 TYPES OF ATTACKS

There are many different types of attacks. A rough classification suggests the following two basic types:

- Attacks at a session layer: these can be attacks that try to alter the flow of BGP messages, for example, the modification, insertion or elimination of messages; attacks that aim to disrupt sessions between two BGP peers; attacks that aim to intercept confidential information between BGP traffic. Among attacks to the session layer, there are those to the TCP protocol (e.g TCP reset, SYN flooding …);
- Denial of Service (or DoS) attacks: these are types of attack that aim to prevent the operation of a service, for example by saturating the resources of a router (memory, CPU).

There are also a number of “unethical” behaviours on the part of net administrators, that can be considered attacks. One of these behaviours, for example, is the concept of “stealing” bandwidth from an Autonomous System (AS), by diverting the traffic between two routers of legitimate AS, using the resources of another (unaware) AS, or by preventing access to certain content, by propagating false routing information.

1.1.1 Denial of Service (DoS) Attacks

Some of these previously cited attacks, e.g. those that lead to the fraudulent disruption of a BGP session, can be considered as attacks that aim to stop an operating service (DoS attacks).

There are many other types of attacks that impede the performance of services offered by BGP. It is not possible to compile an exhaustive list, so for this reason we shall limit ourselves to just a couple of examples.

A first example is the denial of service “reachability of Hosts whose IP addresses belong to a certain IP prefix” (a.k.a. *prefix hijacking*). The idea is very simple. Let’s suppose that we want to hijack the traffic directed to a certain IP prefix, in order to black-hole or analyse it. All that is required is to originate (illegally) the prefix via BGP, and if a BGP peer should choose the sent announcement as the best-path, all traffic towards that prefix would be hijacked from the BGP peer toward the “unethical” router, creating a “black hole”, or a point in the net through which this traffic is lost through lack of valid routing. In the next paragraph we will illustrate this with a couple of examples.

A second example can be found in routers that implement *BGP Route Flap Damping*. By simulating route flaps, it is possible to keep an announcement frozen for a long time, altering correct traffic routing.

Classic DoS type attacks also include those that flood memory resources of a CPU and/or a router. For example, by overwhelming a router with BGP announcements, it is possible to saturate the memory of a router; or, during the three-way-handshake phase of the TCP connection, an attacker could send a huge quantity of TCP SYN (SYN flooding), without ever sending the subsequent TCP Acknowledgement (TCP ACK - namely, the third part of the three-way-handshake), leading to excessive usage of the CPU and ultimately putting the router out of service. These types of attack however, go beyond the argument treated in this document.

1.1.2 *Prefix hijacking*

Prefix hijacking is one of the most common malicious practices. It aims at diverting traffic to a black hole, or even worse, analysing it for unethical ends (e.g. industrial and military espionage).

There have been, and are still, thousands of documented cases of *prefix hijacking*. An historical example is the one described in **this link**, occurred on the 24th February 2008, when Pakistan Telecom (AS 17557) injected an unauthorised BGP announcement of the prefix 208.65.153.0/24, part of the prefix 208.65.152.0/22 assigned by a Regional Internet Registry (RIR), to the well-known YouTube site. One of the Upstream Providers of Pakistan Telecom, PCCW Global (AS3491), then propagated the prefix to the rest of the Internet, allowing Pakistan Telecom to attract part of YouTube traffic on a global scale.

At the root of this, there is an inherent vulnerability within BGP: that is, a lack of checks on the identity of those who propagate the prefixes on the Internet. In other words, there is no way of confirming that the entity from which a prefix originates is authorised to manage it, i.e. that it has legally received this prefix from a *Regional Internet Registry*. An example of prefix hijacking is shown in the following Figure 1.1:

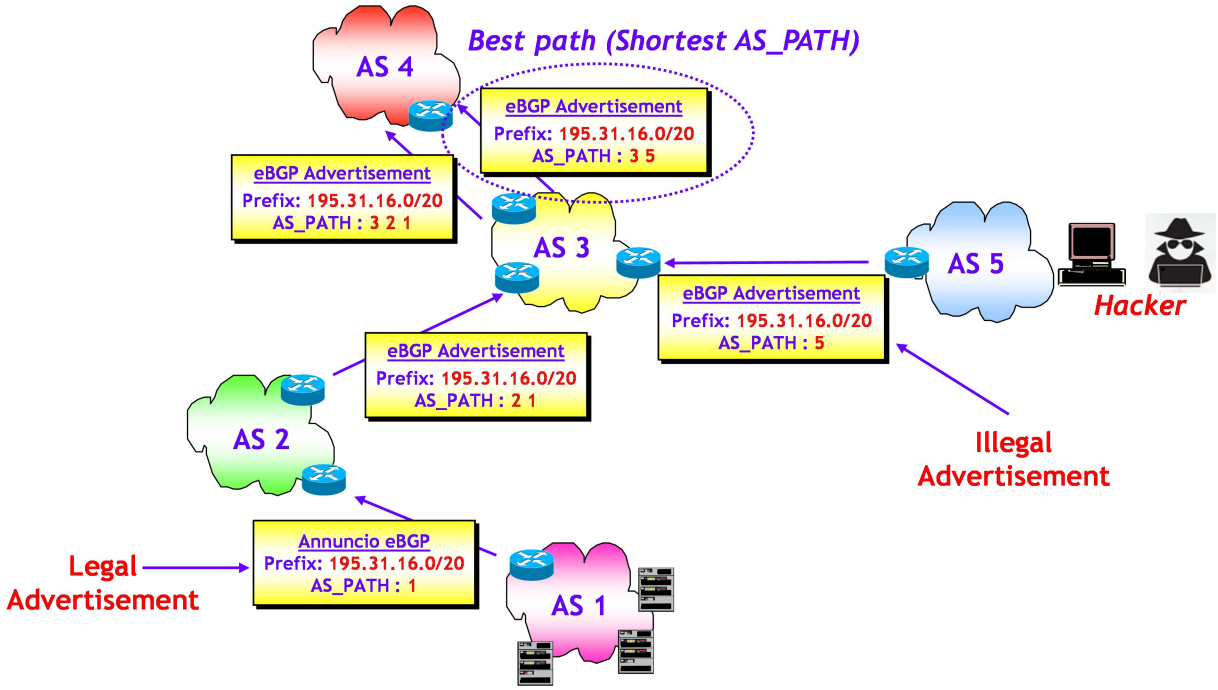


Figure 1.1 – An example of *prefix hijacking*.

AS 1 has received the prefix 195.31.16.0/20 from a *Regional Internet Registry*, and therefore is the only AS authorised to originate it on the Internet. Let us suppose that AS 5, which is unauthorised, either through configuration error or "unethical" behaviour, sends a BGP announcement with the same prefix. A generic AS on the Internet will then receive two announcements of the prefix 195.31.16.0/20, and if the BGP selection process should choose the unauthorised (illegal) announcement as the best path, the traffic directed towards this prefix would be hijacked into a black-hole, and potentially analysed. For example, AS 4 in the figure, because of the shortest AS_PATH (considering equal Local Preferences, as often happens in these cases) would choose the announcements with AS_PATH = [3 5] as best path.

This is not the only type of *prefix hijacking*, though it is one of the most common. Another example of it is when either by an temporary configuration error or by "unethical" behaviour, an unauthorised AS originates a subnet of a prefix that a *Regional Internet Registry* has assigned to a different AS. This type of *prefix hijacking* is precisely the one that caused the incident with YouTube cited earlier. The following Figure 1.2 shows an example.

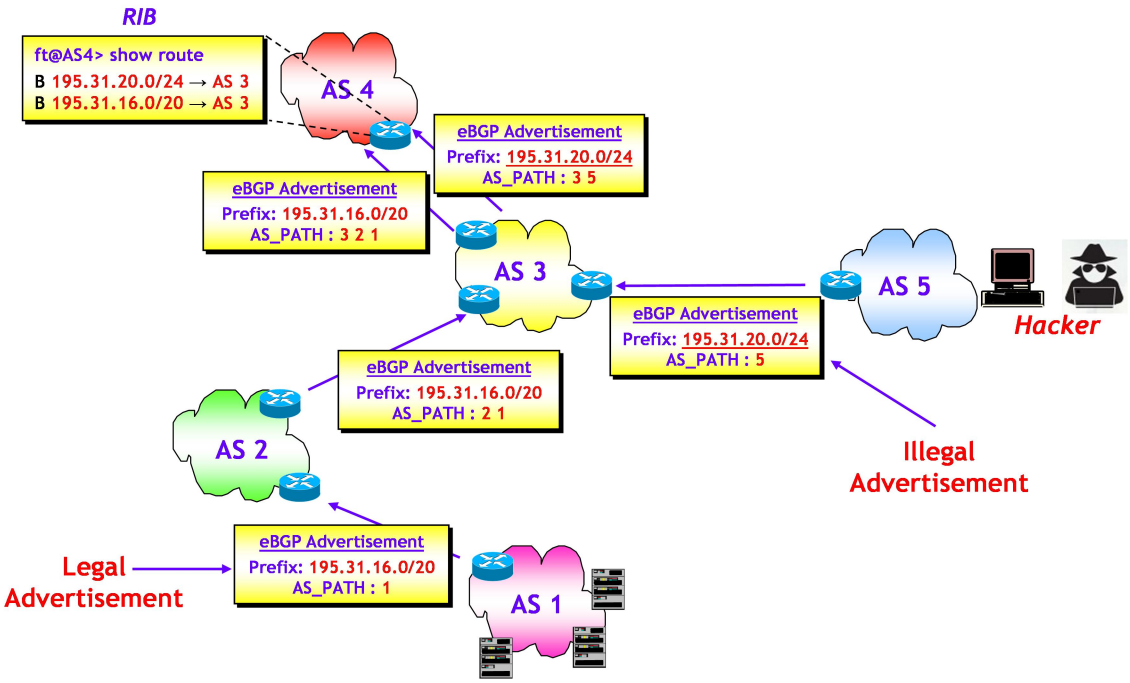


Figure 1.2 – Another example of *prefix hijacking*.

As in the previous example, AS 1 has received prefix 195.31.16.0/20 from a *Regional Internet Registry*, and therefore is uniquely authorised to originate it to the Internet.

Some subnets of this prefix, with prior authorization from AS 1, could be originated from any other AS, but an AS could still originate a subnet without the necessary authorisation.

For example, in the figure, AS 5, without authorization originates subnet 195.31.20.0/24. An AS on the Internet, for example AS 4 in the figure shown, will thus receive two BGP announcements, one with the whole prefix 195.31.16.0/20, originating from AS 1, and one of the subnet 195.31.20.0/24, originating from AS 5. The BGP process considers these two prefixes as different and therefore will choose the best path so that both will go into the IP Routing Table (Routing Information Base or RIB) of the AS 4 interconnecting router. According to the *Longest Match Prefix* rule, a portion of traffic directed to prefix 195.31.16.0/20, the one travelling towards subnet 195.31.20.0/24, will be diverted to AS 5, ending once again in a black-hole.

At the root of the problem once again, is the fact that BGP does not check the identity of who announces the prefixes to the Internet. The only available tool to counter this was the implementation of filters, which is not, unfortunately, a well established practice, and will massively increase the complexity of configurations, without actually solving the problem. For example, how can an intermediate AS know if a remote AS is authorised or not to originate a certain prefix?

The only viable solution for this problem is to set up an authorization system, run by one or more central bodies (e.g. the *Regional Internet Registries*), which by means of digital certification, will allow relatively accurate verification of an AS authorization to originate any prefix to the Internet.

These aspects are the object of the *BGP Resource Public Key Infrastructure* (or *BGP RPKI*) architecture which we will develop in more detail within this document. Before moving onto theory and practice, however, let us first illustrate the fundamental concepts.

1.2 BGP RPKI ARCHITECTURE - INTRODUCTION

Many incidents in the Internet are caused by the propagation of incorrect routing information. The most common threats (or errors), for example *prefix hijacking* or *route leaks*, take advantage of the basic vulnerability of BGP: its inability to verify which Autonomous Systems propagating the announcements are legitimately permitted to do so. In fact, information on the properties of Internet resources (such as AS numbers and prefixes) is contained within public databases know as *Internet Routing Registries* (IRR), some hosted by *Regional Internet Registries* but however maintained by users. It is not therefore within the BGP protocol.

NOTE: on the definition and type of route leaks, the interested reader may consult RFC 7908 - *Problem Definition and Classification of BGP Route Leaks*.

Just to give you an idea of the scale of this phenomenon, in 2018 around 12,600 incidents were documented, affecting 4.4% of AS globally. Around 3,000 AS fell victim of at least one incident and around 1,300 have caused at least one incident (data from ISOC, based on the list of events in **BGPStream.com**).

BGP has no means by which to check if an AS announcing a prefix to the Internet is authorised to do so, namely whether the prefix announced has been effectively assigned to it by a *Regional Internet Registry* or not. Given that every prefix could be announced by or orginated from any AS, independently of its right to do so, an out-of-band mechanism is required to help the BGP check which AS is entitled to originate which prefix.

This mechanism does exist. It is part of the IRR system. As pointed out earlier, these public databases exist, containing information on the properties of prefixes, some run by large organisations and others run by the *Regional Internet Registries*. It is by now common practice to generate filters using information present in the IRR. However there is a limit to this system: can we trust this information? Unfortunately the IRR system is far from complete, as the objects representing the information <*prefix*; *authorised AS*> can be untrustworthy, frequently because content is not updated or contains errors.

To avoid this problem, the *Secure Inter Domain Routing* (or SIDR) group of the *Internet Engineering Task Force* (IETF) in 2012 developed a standard architecture, **RFC 6481** - *A Profile for Resource Certificate Repository Structure* based on a public structure (*Resource Public Key Infrastructure* or RPKI) with distributed databases (*RPKI repositories*) containing the associations <*prefix*; *authorised AS*>.

Each association is linked to a Digital Certificate which allows anyone consulting the repository to check that this association is correct. Each of these declarations (prefixes, AS number and digital certificates) is called *Route Origin Authorization* or ROA.

RPKI uses the format of X.509 digital certificates with extensions for IP addresses and the AS numbers **RFC 3779** - *X.509 Extensions for IP Addresses and AS Identifiers*. The certificates do not include identification information since their scope is simply to transfer usage rights to Internet resources. The RIRs have the role of Trust Anchor and are tasked to issue the digital certificates to their members. The main use of these digital certificates is to validate public keys and the legitimacy of an AS to inject a particular block of prefixes into the BGP, and to use a particular AS number.

2 – BGP RPKI: THEORY

After the brief introduction in the previous chapter, we will illustrate in this chapter the main components of the BGP RPKI architecture and how these interact to create a system that will check whether or not an AS is authorised to originate prefixes to the Internet. Since we want this document to be also a practical guide for the implementation of the RPKI architecture, we will also illustrate how to insert the ROAs in the repository of a RIR (in our case RIPE NCC).

2.1 BGP RPKI ARCHITECTURE: BLOCK DIAGRAM

BGP RPKI architecture, whose block diagram is illustrated in the following Figure 2.1, is based upon databases (*RPKI repository*) containing ROA information, which can be directly entered from RIRs or even from the *National Internet Registry* (NIR), or *Local Internet Registries* (LIR), or *Internet Service Provider* (ISP) (verified however by RIR) by means of a defined *Publication Protocol*.

Usually RIRs thus provide a simplified web interface that allows the complexity of digital certificates to remain hidden from the end user, focusing simply on the creation and publication of ROAs.

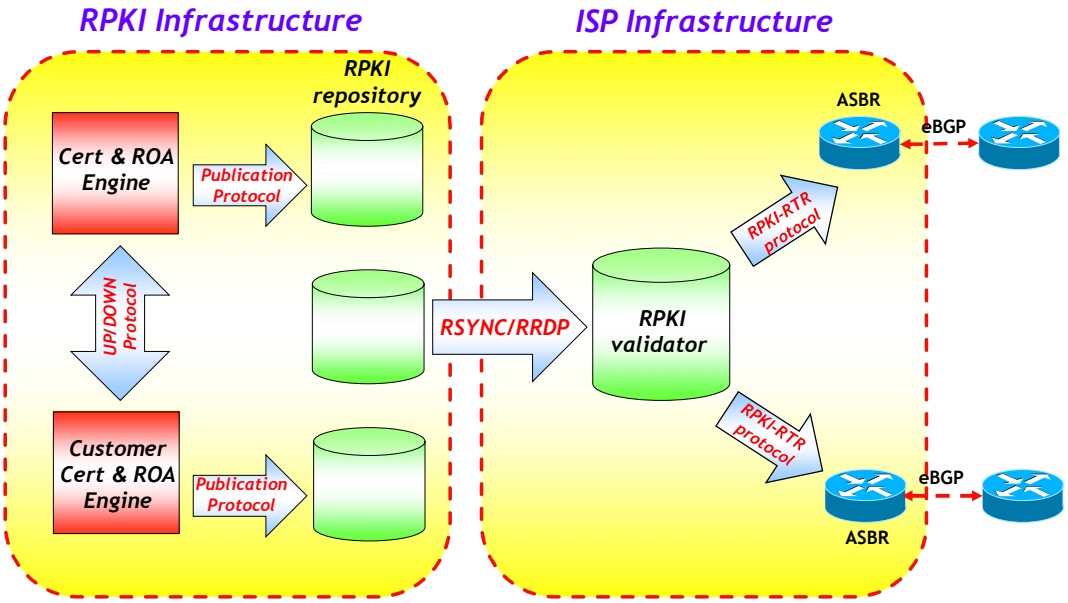


Figure 2.1 – BGP RPKI architecture block diagram.

Each *RPKI repository* contains ROAs, Manifests, public keys and CRLs. On the basis of the ROAs contained in the *RPKI repository*, an AS may validate the BGP announcements received and will select the announcements that it will or will not accept.

In order for this to work, an AS has to be certain that the *RPKI repository* information is correct and synchronized at a global level. RPKI architecture requires a hierarchy of *RPKI repositories* and therefore of digital certificates, which follows the same hierarchy provided by the assignment of IP addresses.

The 5 regional RIRs are responsible for managing each *RPKI repository* for the ROAs in each area and to make this information available to other RIRs and to the ISPs of the rest of the world.

On the ISP side, the architecture will expect an *RPKI Validator* server to be used, which in turn will interface with the *RPKI repositories* of the various RIRs in order to perform local download of ROAs. Edge routers of the ISPs (ASBR), by means of the standard protocol *RPKI-to-Router Protocol* (RFC 6810 - *The Resource Public key Infrastructure (RPKI) to Router Protocol*, January 2013), then download the ROAs present in the *RPKI Validator* locally, inserting them into an *RPKI Table*.

On receipt of a BGP announcement, a router may thus make an inference on the validity of the announcements that it receives, comparing the content of the announcements with that of the ROAs present in the appropriate *RPKI table*. The *RPKI Validators* are usually based upon Open Source software. The best known and most used are the following:

- **Routinator**
- **Ripe NCC Validator**
- **OctoRPKI**
- **Rcynic**
- **FORT**

2.2 RPKI AND PREFIX HIJACKING

The problems of *prefix hijacking* on the Internet have been illustrated in the preceding Chapter 1. Here we would like to show how RPKI architecture can help to avoid incidents of this type.

The following Figure 2.2 shows a classic example of *prefix hijacking* that shows a complete absence of checks on the origin of the announced prefixes (NOTE: in the example we are using IP prefixes and AS numbers from the private range, which is not permitted on the Internet. The concepts however do not change).

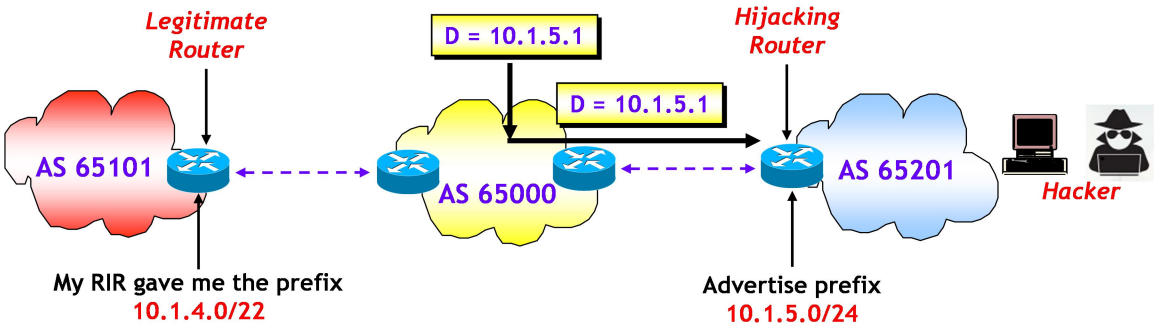


Figure 2.2 – One more example of *prefix hijacking*.

In the figure, AS 65201, without authorisation, originates the subnet 10.1.5.0/24 of the prefix 10.1.4.0/22, which a RIR has assigned to AS 65101. An AS of the Internet, in the figure AS 65000, will thus receive two BGP announcements, one of the whole prefix 10.1.4.0/22, legitimately originating from AS 65101, and one of the more specific subnet 10.1.5.0/24, originating illegally from AS 65201, which is not authorised to originate the prefix 10.1.4.0/22 or any of its subnets.

BGP considers these two prefixes as diverse and therefore selects two best paths, which will both go into the IP routing table of the router of the AS 65000. Using the *Longest Match Prefix* rule, a portion of traffic to prefix 10.1.4.0/22, that is the one directed toward subnet 10.1.5.0/24, will be diverted to AS 65201, ending in a black-hole, or even worse, allowing AS 65201 to spy on this traffic.

At the root of the problem here too, is BGP inability to check the identity of who originates prefixes onto the Internet. As cited in Chapter 1, the only tool available is the use of filters, which is not universally established, complicates the configuration and still does not solve the problem. As an example, how can an intermediate AS know if a remote AS is authorized or not to originate a specific prefix?

The best solution to this problem is to implement RPKI architecture. The following Figure 2.3 refers to the preceding example, but with one main difference that the AS 65000 checks the legitimacy of the received announcement via its *RPKI Validator*.

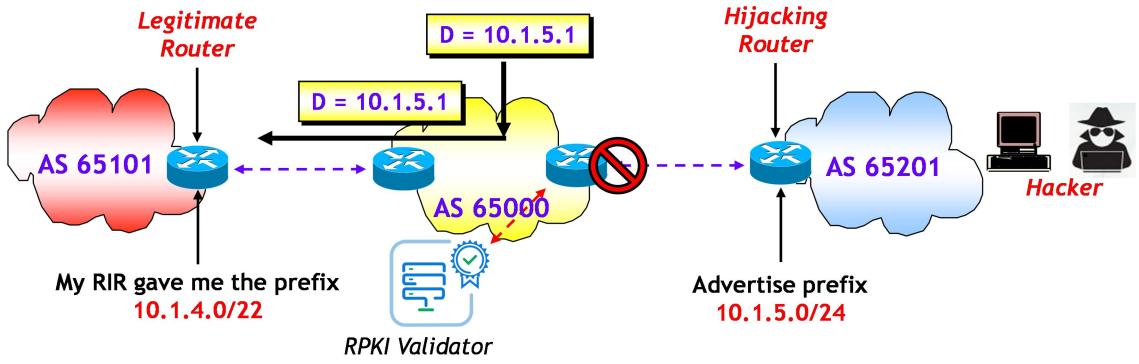


Figure 2.3 – The use of RPLKI to prevent *prefix hijacking*.

Once the *RPKI Validator* declares the received announcement invalid, the AS 65000 router discards the announcement, thus avoiding the problem of *prefix hijacking*. As a matter of fact, AS 65000 could take a different decision, even if discarding the announcement is the most sensible.

2.3 ROUTE ORIGIN AUTHORIZATION (ROA)

ROAs are key elements of BGP RPKI architecture. They may be seen as objects that provide a means of verifiying if an AS is authorised to originate a certain IP prefix and its subnets. Each ROA has 4 components:

- An IPv4 or IPv6 prefix with fixed prefix length. Typically it is a prefix assigned by an RIR to a NIR/LIR/ISP.
- A maximum mask length, that specifies which IP subnets from the originating prefix may be announced.
- The AS number of the AS which is authorized to originate the IP prefix or one of its allowed subnets.
- A digital signature, based on a public / private key system.

The format of ROAs and their usage is described in **RFC 6811** - *BGP Prefix Origin Validation, January 2013*.

The following Figure 2.3 gives a pictorial representation of an IPv4 ROA and of an IPv6 ROA. The IPv4 prefix which is allowed in the Internet is 195.31.0.0/16, and the AS which is authorised to originate the prefix has AS number 12345. The AS 12345 is also authorised to originate all the subnets of the IPv4 prefix 195.31.0.0/16, whose mask length is less than or equal to 24 (e.g. 195.31.1.0/24, 195.31.4.0/22, etc.).

The same is true for the IPv6 prefix.

Originated Prefix	195.31.0.0/16	2a01:20b1:1::/48
Max mask length	24	48
Origin AS	12345	12345
Digital Signature	qç!r5@eX!%89?@cv!	sdg@!dr34@?1QWx!@a

IPv4 *ROA* IPv6 *ROA*

Figure 2.3 – ROA format.

2.3.1 How to create a ROA: instruction for use

In order to create ROAs, the RIRs provide a hosted solution hiding all of the complex cryptographic details. An operator may thus focus their attention solely on the creation and publication of ROAs by announced prefixes. This is a quick guide that explains how to create a ROA on the RIPE NCC portal.

The first thing to do is to access the RIPE NCC portal. Once authenticated, click on the tab "*My Resources*", as shown in the following Figure 2.4.

Figure 2.4 – RIPE NCC portal – private area.

On the next screen, click on the menu to the left on object "*RPKI dashboard*" and then on the tab "*Route Origin Authorisations (ROAs)*".

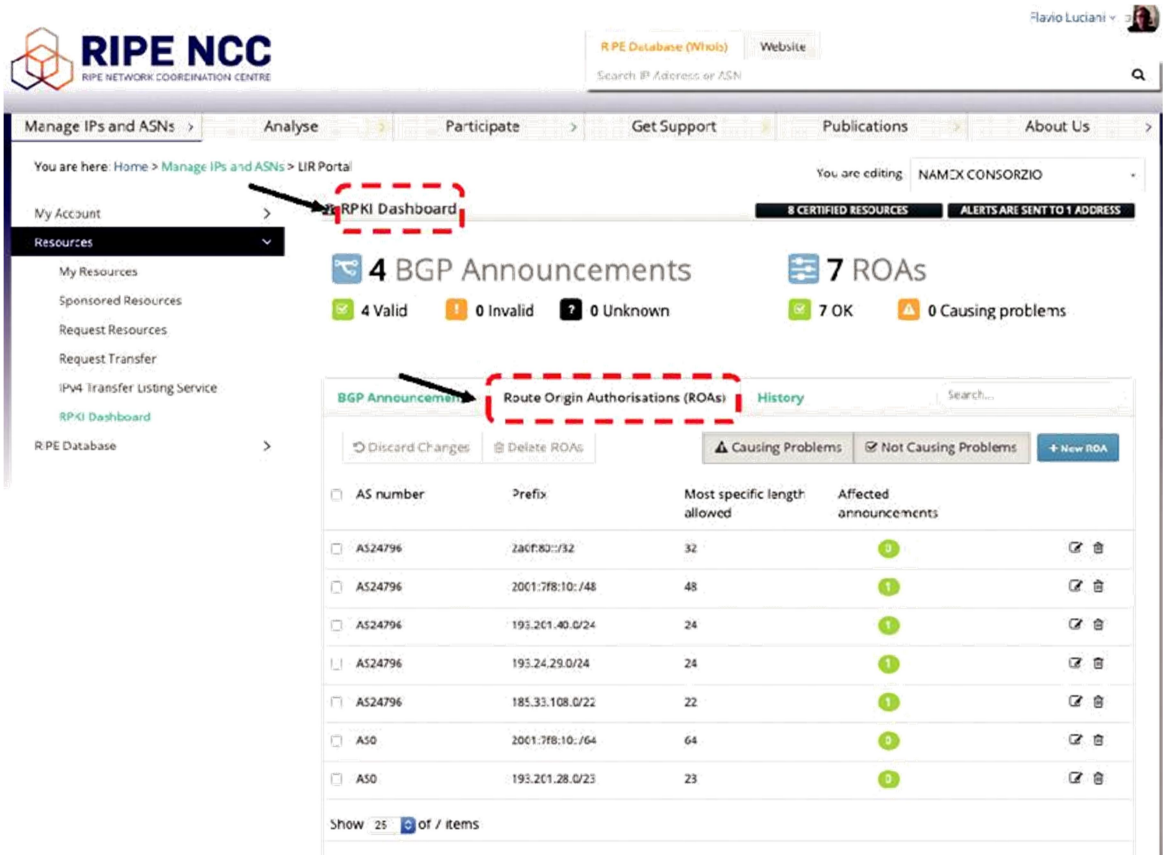


Figure 2.5 – List of ROAs.

From here it is possible to display the list of created ROAs, add new ones or delete one or more ROAs. To create a new ROA, click on the button "+ New ROA", and then enter the ROA components: AS number/Prefix/Max length.

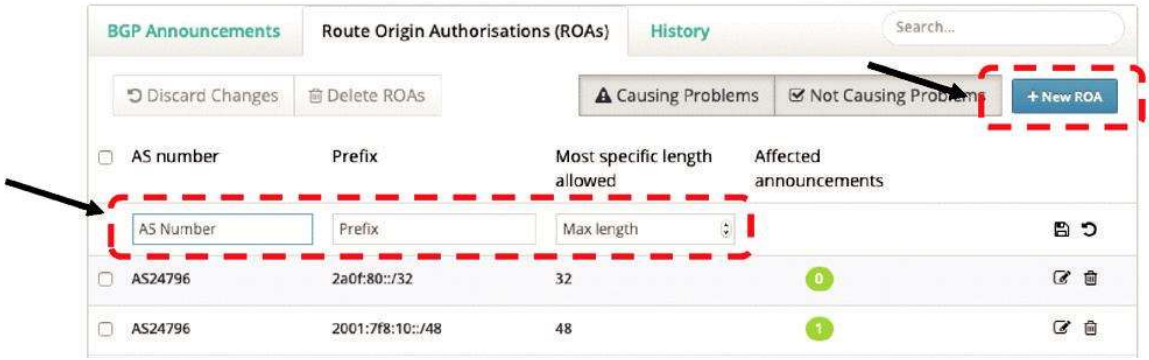


Figure 2.6 – Creating a new ROA.

Game over!!!

2.3.2 ROAs with Origin AS = 0

AS number 0 is indicated in the IANA registries as reserved (see **this link**). **RFC 6491** - *Resource Public Key Infrastructure (RPKI) Objects issued by IANA*, specifies that the value of AS = 0 in an ROA is used to identify prefixes that should not be announced to the Internet and therefore are not usable for packet routing.

This allows whoever has a certain IP prefix to indicate that the prefix and possibly all of the IP subnets, may not be used for IP packet routing.

In the definition of an ROA with AS = 0, good practice defines maximum prefix length as equal to the mask length of the prefix, even if some may prefer to define maximum mask length as 32 for IPv4 and 128 for IPv6. The end result remains the same.

The use of AS = 0 has been specified in **RFC 7607** - *Codification of AS 0 Processing, August 2015*.

2.4 VALIDATION PROCESS

The validation process of a BGP announcement with address-family IPv4 or IPv6 unicast, is based on the comparison between the information contained in the announcement, and the ROAs present in the database of the router itself, downloaded from the RPKI Validator.

There are three possible results from the validation process, as described in RFC 6811: ***Valid***, ***Invalid*** and ***NotFound***. To understand the meaning, let us suppose that we want to validate a BGP announcement of address-family IPv4 or IPv6 unicast, containing the following information:

- NLRI = Pfx/Mask
- Origin AS = AS-X

The possible results of the validation process are as follows:

- ***Valid***: the RPKI Validator contains an ROA = <Pfx-ROA/Mask-ROA; Max-Mask; AS-ROA> with AS-X = AS-ROA, Pfx/Mask is a more specific prefix of Pfx-ROA/Mask-ROA and Mask ≤ Max-Mask.
- ***Invalid***: the RPKI Validator contains an ROA = <Pfx-ROA/Mask-ROA; Max-Mask; AS-ROA>, with Pfx/Mask which is a more specific prefix of Pfx-ROA/Mask-ROA, but either AS-X ≠ AS-ROA and/or Mask > Max-Mask.
- ***NotFound*****:** within RPKI Validator there are no ROAs = <Pfx-ROA/Mask-ROA; Max-Mask; AS-ROA> so that Pfx/Mask is a more specific prefix of Pfx-ROA/Mask-ROA.

The following Figure 2.7 shows an example that better explains the validation statuses of a BGP announcement. The announcements are compared with an ROA characterised by:

- Pfx-ROA/Mask-ROA = 79.140.80.0/20
- Max-Mask = 24
- AS-ROA = 6762

With respect to this ROA, the first of the two announcements have been deemed ***Valid***, since both originate from AS 6762, and both contain an IPv4 prefix which is subnet of the "root" prefix 79.140.80.0/20 and they both have mask length, respectively 21 and 22, less than the value of *Max-Mask* of the ROA, equal to 24.

The third announcement, prefix 79.140.84.0/22 originating from AS 2914, is deemed ***Invalid*** since it is a subnet of prefixes 79.140.80.0/20 with a mask length less than the value of *Max-Mask* of the ROA, but originating from a different AS (= 2914) than that of the ROA (*AS-ROA* = 6762).

The fourth announcement, prefix 79.140.91.0/25 originating from AS 6762, is also deemed ***Invalid*** since it is a subnet of prefixes 79.140.80.0/20 orginating from the same AS present in the ROA (*AS-ROA* = 6762), but with a mask length (=25) greater than the value of *Max-Mask* of the ROA.

Finally, the last announcement, that of prefix 79.140.0.0/16 orginating from AS 1299, is designated as ***NotFound*** since it is not a subnet of prefix 79.140.80.0/20. It would have been deemed ***NotFound*** even if the announcement had been announced by AS 6762.

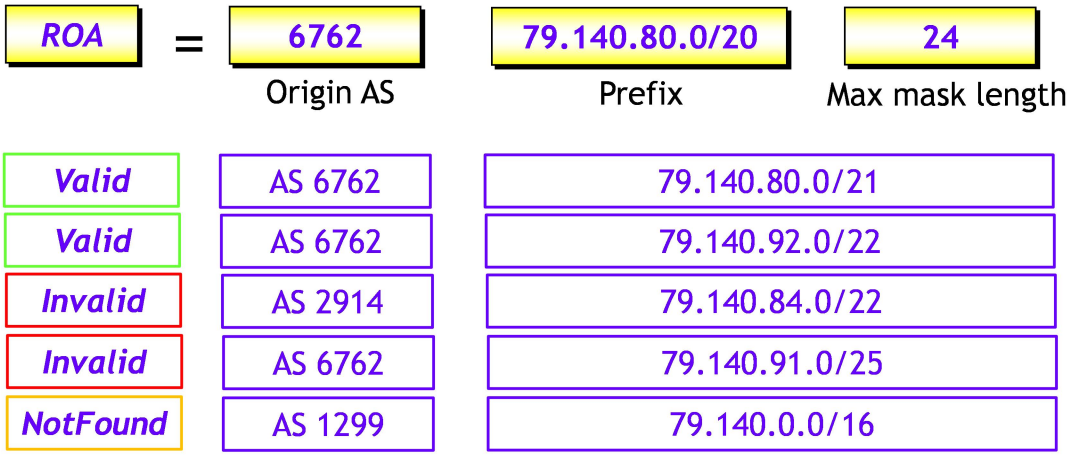


Figure 2.7 – An example of ROA validation process.

NOTE: In the case of ROAs with *AS-ROA* = 0, the validation process will always result in a designation of ***Invalid***.

Here we end the poetry (the theory!) and it is therefore time to move onto the prose (practical implementation!).

3 – BGP RPKI: FROM THEORY TO PRACTICE

Having surveyed the main elements of RPKI architecture and its basic functionalities, it is now time to put the theory into practice. In this last chapter we will illustrate RPKI implementation both in the Cisco (IOS XE) and in the Juniper (JUNOS) environments.

NOTE: In the following scenario the AS numbers used were taken randomly (we apologize to their legitimate owners). The prefixes announced by each AS were taken from *RPKI repository* of RIPE NCC, in order to verify the validity/non-validity or the inability to validate their status.

The following Figure 3.1 shows the topology of a Case Study which we will use to show how to implement RPKI architecture and to check how it works:

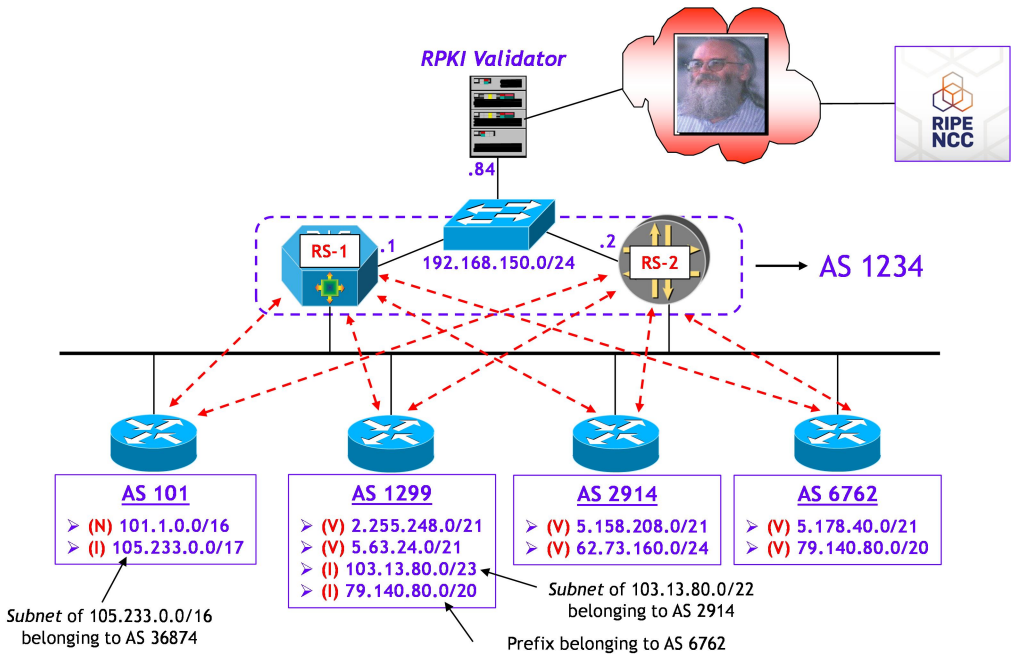


Figure 3.1 – Network scenario for RPKI implementation.

The scenario shows an IXP (AS 1234) providing a BGP RPKI validation service to its own members (exemplified in this case study by AS 101, 1299, 2914 and 6762) via its *Route Servers* (RS-1 and RS-2).

In order to achieve this, the two Route Servers use the *RPKI Validator* provided by RIPE NCC, which is the RIR for Europe, the Middle East and parts of Central Asia, installed on a COTS server, with the IP address 192.168.150.84. The TCP port used by the RIPE NCC *RPKI Validator* is 8282. The *RPKI Validator* is in turn linked via the Internet to the RIPE NCC *RPKI repository*.

The Route Servers used here consist of two routers, a Cisco router (RS-1) and a Juniper router (RS-2) that run respectively the Cisco IOS XE and Juniper JUNOS operating systems. Each AS announces the prefixes shown in the figure. To make the case study more interesting, we have also announced two prefixes belonging to two other AS (see the figure) to AS 101 and 1299, and which are therefore not valid. In the figure we show the validation statuses that we would expect to find after the validation process (**V**=***Valid***; **N**= ***NotFound***; **I**=***Invalid***).

NOTE: in an IXP infrastructure, prefix validation on Route Servers is advised by the **MANRS** (**M**utually **A**greed **N**orms for **R**outing **S**ecurity) recommendations, which quote on the page dedicated to IXPs "*Action 1. Prevent propagation of incorrect routing information. (Mandatory)*": "**IXPs using Route Server to facilitate multilateral peering should use it to validate received route announcements from a peer and subsequently filter them to other peers.**"

3.1 CONFIGURATIONS ON CISCO PLATFORMS

The configurations on a Cisco router in order to connect it to an *RPKI Validator* are very simple, you only need to specify three parameters:

- IP address of the RPKI Validator.
- TCP port to use. Usually the port used by the RPKI Validator is 323, but the RPKI Validator provided by RIPE NCC uses port 8282.
- The query time of the RPKI Validator (refresh time) for the download of the ROAs (validated cache). A typical value used in practical applications could be 600 seconds.

The configurations required are as follows:

```
router(config)# router bgp AS-number
router(config-router)# bgp rpki server tcp IP-RPKI-Validator
                          port port-RPKI-Validator refresh seconds
```

If for reliability reasons there are two or more *RPKI Validators*, it is enough to simply repeat the command "**bgp rpki** ...", varying the IP address of the server which houses the *RPKI Validator*.

The configuration described above is applicable to a Cisco router which runs classic IOS or IOS XE. For reasons of completeness we will also show the configurations in routers that run IOS XR.

```
RP/0/RP0/CPU0:router(config)# router bgp AS-number
RP/0/RP0/CPU0:router(config-bgp)# rpki server IP-RPKI-Validator
RP/0/RP0/CPU0:router(config-bgp-rpki-server)# transport
                                      tcp port port-RPKI-Validator
RP/0/RP0/CPU0:router(config-bgp-rpki-server)# refresh-time seconds
```

The use of RPKI validation of BGP announcements modifies the BGP selection process, which no longer considers all announcements (of a single prefix) in order to determine the best-path. In fact, ***Invalid*** announcements are not used (or at least, should not be used) in the BGP selection process. An ***Invalid*** announcement will therefore never enter in a Routing Table, which is, ultimately, the main purpose of the RPKI architecture.

It is however possible for ***Invalid*** announcements to participate in the BGP selection process, for example in the testing phase, when one doesn't feel (yet) comfortable to drop invalids, using the following commands:

IOS/IOS XE

```
router(config)# router bgp AS-number
router(config-router)# address -family ipv4 unicast
router(config-router-af)# bgp bestpath prefix-validate
                                                allow-invalid
```

IOS XR

```
RP/0/RP0/CPU0:router(config)# router bgp AS-number
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# bgp bestpath origin-as
                                                    allow invalid
```

Please note however, that in the presence of one or more announcements from the same IP prefix, of which at least one is ***Valid***, the ***Valid*** advertisement will take precedence and will be elected best-path independently of the common BGP metrics.

For example, let us suppose we have two BGP announcements of prefix 79.1.140.80.0/20, the first ***Valid*** and the second ***Invalid***, and let us also suppose that the first has *Local Preference* = 100 and the second, *Local Preference* =200. Without applying RPKI validation process, the second advertisement would be chosen best-path, with RPKI validation process enabled, the first announcement would become best-path since it is ***Valid***.

Sometimes, to facilitate testing, it can be useful to completely ignore information on validation statuses during the selection process. This can be achieved using the following commands:

IOS/IOS XE

```
router(config)# router bgp AS-number
router(config-router)# address-family ipv4 unicast
router(config-router-af)# bgp bestpath prefix-validate disable
```

IOS XR

```
RP/0/RP0/CPU0:router(config)# router bgp AS-number
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# bgp origin-as validation disable
```

These commands, whose usage should be temporary and for specific situations, will disable the use of information on validation statuses in the selection process, but will not disconnect the router from the *RPKI Validator*.

NOTE: it is important to be aware that using these commands either to disable the use of information on validation statuses in the selection process, or allowing the use of ***Invalid*** announcements in the selection process, is not considered a good practice, since it undermines the very foundations of the use of RPKI architecture.

Finally, the results of the validation process can be used to apply policies that perform actions on announcements, depending on validation status. These policies may be configured via classic “route-map” tools (IOS and IOS XE) and “route-policy” (IOS XR). For these, new “match” conditions have been defined that refer to validation statuses.

We give below an example of IOS/IOS XE environments in which ***Valid*** announcements are assigned a *Local preference* = 200, ***NotFound*** announcements are assigned a *Local Preference* = 100, and ***Invalid*** announcements are assigned a *Local Preference* = 50. Note that in order to execute actions on ***Invalid*** announcements, the command shown above should be used, permitting these announcements to participate in the BGP selection process.

```
route-map RPKI-POLICY permit 10
  match rpki invalid
  set local-preference 50
!
route-map RPKI-POLICY permit 20
  match rpki not-found
  set local-preference 100
!
```

```
route-map RPKI-POLICY permit 30
  match rpki valid
  set local-preference 200
!
router bgp AS-number
  bgp bestpath prefix-validate allow-invalid
  neighbor 10.1.1.1 route-map RPKI-POLICY in
```

Below is a second example, but for Cisco platforms using IOS XR. The only difference compared to the one just shown is that in this case ***Invalid*** announcements are discarded.

```
route-policy RPKI-POLICY
 if validation-state is valid then
   set local-preference 200
 elseif validation-state is not-found then
   set local-preference 100
 else
   drop
!
router bgp AS-number
  neighbor 10.1.1.1
    address-family ipv4 unicast
      route-policy RPKI-POLICY in
```

With RPKI enabled, the default behaviour of a Cisco router is not to propagate the RPKI validation statuses of received eBGP announcements on iBGP sessions. To allow propagation, the following commands should be used:

IOS / IOS XE

```
router (config) # router bgp  AS-number
router (config-router) # address-family ipv4 unicast
router (config-router-af) # neighbor IP-iBGP-peer announce rpki state
router (config-router-af) # neighbor IP-iBGP-peer send-community
                                                          extended
```

IOS XR

```
RP/0/RP0/CPU0: router (config) # router bgp  AS-number
RP/0/RP0/CPU0: router (config-bgp) # address-family ipv4 unicast
RP/0/RP0/CPU0: router (config-bgp-af) # bgp origin as validation
                                                  signal ibgp
```

Note that the information on the validation status is propagated using an appropriate BGP *Extended Community* . The format of the *Extended Community* is as follows:

- 1st byte = 0x43 - indicates *Extended Community* of the type *Non-Transitive Opaque* (see RFC 7153, Sec. 5.2.9).
- 2nd byte = 0x0 - indicates *BGP Origin Validation State* (see RFC 7153 , Sec. 5.2.9).
- 3rd-8th bytes = 0x0/0x1/0x2 respectively for the states ***Valid/NotFound/Invalid*** (see RFC 8097 , sect. 2).

Upon receipt of an iBGP advertisement containing the *BGP Extended Community*, the *iBGP peer* is able to infer the validation state of the advertisement, without the need for a connection to the *RPKI Validator*.

3.2 CONFIGURATIONS ON JUNIPER PLATFORMS

The configurations required for a Juniper router running JUNOS are a little more complex (even the basic ones) and require more commands with respect to the Cisco equivalent. However they do allow considerable freedom in the management of ***Valid*/*NotFound*/*Invalid*** announcements. This additional freedom derives from the fact that using JUNOS to implement the RPKI architecture has as a mandatory requirement, the use of a *routing policy*, through which it is possible to decide whether or not to accept certain types of announcements, to associate some BGP Community with validation statuses, certain Local preference values, and so forth.

The configuration is divided into two parts. In the first part a link is established with the server where the *RPKI Validator* resides. To this end, the commands to execute are the following:

```
[edit routing-options]
validation {
    group name {
        session IP-address-RPKI-Validator {
            refresh-time seconds;
            hold-time seconds;
            port port-RPKI-Validator;
            local-address local-IP-address;
        }
    }
}
```

Within each group there may be at most two sessions to two different RPKI Validators. If compared to the Cisco configurations, another timer may also appear, the Hold Time, which is the period of time after which the TCP session is reset, following zero activity with the RPKI Validator. This timer must of course be greater than the refresh-time.

NOTE: in some older versions of JUNOS, a specific value of Hold Time was not mandatory. In the version used in our Case Study (17.2R1.13) it was mandatory.

The second part of the configuration focuses on checking validation and on the policies applied to announcements, depending on the outcomes of the validation process. These policies are quite flexible, thus allowing considerable freedom. Moreover, within these same policies it is possible to assign values of Extended Community as mentioned earlier, that may be used in iBGP sessions to propagate validation statuses of announcements.

The generic policy configuration is as follows:

```
[edit policy-options]
policy-statement Name-RPKI-Policy {
    term VALID {
        from {
            protocol bgp;
            validation-database valid;
        }
        then {
            validation-state valid;
            community add COMM-V;
            accept;
        }
    }
```

```
    term INVALID {
        from {
            protocol bgp;
            validation-database invalid;
        }
        then {
            validation-state invalid;
            community add COMM-I;
            accept | reject;
        }
    }
    term NOTFOUND {
        from {
            protocol bgp;
            validation-database unknown;
        }
        then {
            validation-state unknown;
            community add COMM-U;
            accept | reject;
        }
    }
}
community COMM-I members 0x4300:2;
community COMM-U members 0x4300:1;
community COMM-V members 0x4300:0;
```

NOTE: in theory it is possible to also use the "reject" clause for ***Valid*** announcements, but we have not shown this option, since it makes little sense from a practical point of view.

This routing-policy will then be applied to the BGP process in the "import" direction, on all eBGP sessions for which validation of the received prefixes is required:

```
[edit protocols bgp]
group eBGP {
    import Name-RPKI-Policy ;
. . .
}
```

Now all that remains in our discussion, is to apply all configurations seen thus far to our Case Study, and to examine the main "show" commands that will verify whether our RPKI architecture works has expected.

3.3 CASE STUDY

Our Case Study is based on the scenario described at the beginning of this chapter. The configurations of RPKI architecture executed on the two Route Servers are as follows:

RS-1 (Cisco IOS XE)

```
router bgp 1234
  bgp rpki server tcp 192.168.150.84 port 8282 refresh 600
```

RS-2 (Juniper JUNOS)

```
[edit routing-options]
validation {
    group RPKI-VAL {
        session 192.168.150.84 {
            refresh-time 600;
            hold-time 1500;
            port 8282;
            local-address 192.168.150.2;
        }
    }
}

[edit policy-options]
policy-statement VALIDATION {
    term VALID {
        from {
            protocol bgp;
            validation-database valid;
        }
        then {
            validation-state valid;
            community add COMM-V;
            accept;
        }
    }
    term INVALID {
        from {
            protocol bgp;
            validation-database invalid;
        }
        then {
            validation-state invalid;
            community add COMM-I;
            reject;
        }
    }
    term NOTFOUND {
        from {
            protocol bgp;
            validation-database unknown;
        }
        then {
            validation-state unknown;
            community add COMM-U;
            accept;
        }
    }
}
community COMM-I members 0x4300:2;
community COMM-U members 0x4300:1;
community COMM-V members 0x4300:0;

[edit protocols bgp]
group eBGP {
    import VALIDATION;
. . .
}
```

Please note that in the configuration of RS-2, the VALIDATION routing policy will discard all ***Invalid*** announcements. What happens in reality, as we will see shortly, is that JUNOS places them in a hidden part of the Routing Table.

Now let's check whether these configurations can actually "do their job". The first check is to see whether the TCP connection to the *RPKI Validator* is active. We will first investigate RS-1:

```
RS-1# show bgp ipv4 unicast rpki servers
BGP SOVC neighbor is 192.168.150.84/8282 connected to port 8282
Flags 64, Refresh time is 600, Serial number is 133, Session ID is
4568
InQ has 0 messages, OutQ has 0 messages, formatted msg 462
Session IO flags 3, Session flags 4008
 Neighbor Statistics:
  Prefixes 86361
  Connection attempts: 12
  Connection failures: 1
  Errors sent: 0
  Errors received: 0
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Mininum incoming TTL 0, Outgoing TTL 255
Local host: 192.168.150.1, Local port: 13642
Foreign host: 192.168.150.84, Foreign port: 8282
Connection tableid (VRF): 0
Maximum output segment queue size: 50

. . . < rest of the output omitted for brevity > . . .
```

The results displayed show that the TCP connection has been correctly established (**Connection state is ESTAB**) and that the 4 components that identify the TCP connection (IP addresses and the source/destination ports) are:

- **Local host: 192.168.150.1, Local port: 13462**
- **Foreign host: 192.168.150.84, Foreign port: 8282**

Furthermore other information of lesser importance is displayed. Amongst this information is shown the number of downloaded ROAs (**Prefixes 86361**). Note that the display has been shortened for brevity.

Similarly, verification for RS-2 may be obtained with a similar display command:

```
ft@RS-2> show validation session detail
Session 192.168.150.84, State: up, Session index: 2
  Group: RPKI-VAL, Preference: 100
  Local IPv4 address: 192.168.150.2, Port: 8282
  Refresh time: 600s
  Hold time: 1500s
  Record Life time: 3600s
  Serial (Full Update): 133
  Serial (Incremental Update): 133
    Session flaps: 0
    Session uptime: 23:44:23
    Last PDU received: 00:00:13
    IPv4 prefix count: 75112
    IPv6 prefix count: 11249
```

The following commands will display the IPv4/IPv6 *RPKI Table*, or rather the list of ROAs that the router downloads from the *RPKI Validator* and enters into local memory. Since the ROAs list is very long, say, in the order of tens of thousands and in the future (one hopes) in the order of hundreds of thousands, it makes sense to only display the area of interest.

For example, suppose we want to display on local RS-1 *RPKI Table* all of the ROAs IPv4/IPv6 from AS 6762. The commands would be as follows:

```
RS-1#show bgp ipv4 unicast rpki table | i 6762
5.178.40.0/21          21      6762        0       192.168.150.84/8282
45.189.119.0/24        24      6762        0       192.168.150.84/8282
79.140.80.0/20         20      6762        0       192.168.150.84/8282
89.221.32.0/20         20      6762        0       192.168.150.84/8282
93.186.128.0/21        21      6762        0       192.168.150.84/8282
93.186.136.0/22        22      6762        0       192.168.150.84/8282
149.3.176.0/21         21      6762        0       192.168.150.84/8282
176.115.184.0/22       22      6762        0       192.168.150.84/8282
185.70.200.0/22        22      6762        0       192.168.150.84/8282
185.100.112.0/22       22      6762        0       192.168.150.84/8282
195.22.192.0/19        19      6762        0       192.168.150.84/8282
213.144.160.0/19       19      6762        0       192.168.150.84/8282

RS-1#show bgp ipv6 unicast rpki table | i 6762
2001:41A8::/32         32      6762        0       192.168.150.84/8282
2801:15:7000::/48      48      6762        0       192.168.150.84/8282
```

The first “show…” command shows the *RPKI Table* for IPv4 ROAs, while the second the RPKI Table for IPv6 ROAs. The first column contains the prefix of Validated ROA, the second column contains the maximum mask length and the third column shows the Origin AS. The fourth column is always empty and the last column contains the IP address of the *RPKI Validator* (=192.168.150.84) and the TCP port (=8282).

To display the same portion of the RPKI Table on RS-2, use the following command:

```
ft@RS-2> show validation database | match 6762
5.178.40.0/21-21         6762 192.168.150.84              valid
45.189.119.0/24-24       6762 192.168.150.84              valid
79.140.80.0/20-20        6762 192.168.150.84              valid
89.221.32.0/20-20        6762 192.168.150.84              valid
93.186.128.0/21-21       6762 192.168.150.84              valid
93.186.136.0/22-22       6762 192.168.150.84              valid
149.3.176.0/21-21        6762 192.168.150.84              valid
176.115.184.0/22-22      6762 192.168.150.84              valid
185.70.200.0/22-22       6762 192.168.150.84              valid
185.100.112.0/22-22      6762 192.168.150.84              valid
195.22.192.0/19-19       6762 192.168.150.84              valid
213.144.160.0/19-19      6762 192.168.150.84              valid
2001:41a8::/32-32        6762 192.168.150.84              valid
2801:15:7000::/48-48     6762 192.168.150.84              valid
```

At this point what is left is to check the validation status of the eBGP announcements. On RS-1 we have:

```
RS-1#show ip bgp
. . .
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
V*>  2.255.248.0/21   10.0.0.2                                0 1299 i
V*>  5.63.24.0/21     10.0.0.2                                0 1299 i
V*>  5.158.208.0/21   10.0.0.3                                0 2914 i
V*>  5.178.40.0/21    10.0.0.4                                0 6762 i
V*>  62.73.160.0/24   10.0.0.3                                0 2914 i
V*>  79.140.80.0/20   10.0.0.4                                0 6762 i
I*                    10.0.0.2                     200        0 1299 i
N*>  101.1.0.0/16     10.0.0.1                                0 101 i
N*>  103.13.80.0/23   10.0.0.2                                0 1299 i
I*   105.233.0.0/17   10.0.0.1                     200        0 101 i
```

We will leave to the reader to check the correctness of the validation statuses, by comparing the announcements with the scenario described in the Figure 3.1. An interesting point of note regards the announcements of prefix 79.140.80.0/20, originating (legally) from AS 6762 and (illegally) from AS 1299. Of these two announcements, one has ***Valid*** status and the other ***Invalid*** status. Despite the latter having a higher Local Preference (=200), the BGP selection process will always choose the ***Valid*** status announcement as best-path.

This would not change even if ***Invalid*** announcements were permitted in the selection process using the following command:

```
router bgp 1234
 address-family ipv4
  bgp bestpath prefix-validate allow-invalid
```

To conclude, let us examine the BGP announcements on RS-2 (Note: in this case we have left all Local Preferences at default values).

```
ft@RS-2> show route table inet.0 protocol bgp

inet.0: 16 destinations, 17 routes (15 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

2.255.248.0/21     *[BGP/170] 15:22:08, localpref 100
                      AS path: 1299 I, validation-state: valid
                    > to 10.0.0.2 via ge-0/0/1.0
5.63.24.0/21       *[BGP/170] 15:22:08, localpref 100
                      AS path: 1299 I, validation-state: valid
                    > to 10.0.0.2 via ge-0/0/1.0
5.158.208.0/21     *[BGP/170] 15:22:08, localpref 100
                      AS path: 2914 I, validation-state: valid
                    > to 10.0.0.3 via ge-0/0/1.0
5.178.40.0/21      *[BGP/170] 15:22:08, localpref 100
                      AS path: 6762 I, validation-state: valid
                    > to 10.0.0.4 via ge-0/0/1.0
62.73.160.0/24     *[BGP/170] 15:22:08, localpref 100
                      AS path: 2914 I, validation-state: valid
                    > to 10.0.0.3 via ge-0/0/1.0
79.140.80.0/20     *[BGP/170] 00:01:19, localpref 100
                      AS path:  6762 I, validation-state: valid
                    > to 10.0.0.4 via ge-0/0/1.0
```

```
101.1.0.0/16        *[BGP/170] 15:22:08, localpref 100
                      AS path: 101 I, validation-state: unknown
                    > to 10.0.0.1 via ge-0/0/1.0
103.13.80.0/23      *[BGP/170] 15:22:08, localpref 100
                      AS path: 1299 I, validation-state: unknown
                    > to 10.0.0.1 via ge-0/0/1.0
```

As you can see, the table only shows the announcements which have ***Valid*** and ***NotFound*** (which JUNOS calls ***Unknown***) status. So what about ***Invalid*** announcements? They may be found in the "hidden" part of the table (**2 hidden**). To display, use the following command:

```
ft@RS-2> show route table inet.0 hidden

inet.0: 16 destinations, 17 routes (15 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

79.140.80.0/20       [BGP ] 11:18:49, localpref 100
                      AS path: 1299 I, validation-state: invalid
                    > to 10.0.0.2 via ge-0/0/1.0
105.233.0.0/17       [BGP ] 11:18:49, localpref 100
                      AS path: 101 I, validation-state: invalid
                    > to 10.0.0.1 via ge-0/0/1.0
```

For those of you unfamiliar with JUNOS, we point out that "hidden" announcements do not participate in the BGP selection process and therefore cannot become best-path.

NOTE: the behaviour of JUNOS in the BGP selection process differs to that of Cisco in that it does not account for prefix validation statuses. For example, by accepting Invalid prefixes in the policy and defining the same Local Preferences of the prefix 79.140.80.0/20 as per RS-1, the selection process will choose the ***Invalid*** announcement as best-path, as shown here:

```
ft@RS-2> show route 79.140.80.0/20

inet.0: 16 destinations, 17 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
79.140.80.0/20      *[BGP/170] 00:05:38, localpref 200
                      AS path: 1299 I, validation-state: invalid
                    > to 10.0.0.2 via ge-0/0/1.0
                     [BGP/170] 09:43:30, localpref 100
                      AS path: 6762 I, validation-state: valid
                    > to 10.0.0.4 via ge-0/0/1.0
```

Therefore to avoid any situation contrary to the spirit of RPKI architecture, we suggest as best practice to always reject ***Invalid*** advertisements.

END OF THE STORY!!!

We hereby conclude this short essay on the RPKI architecture. We hope we have been able to demonstrate the importance of applying it to real networks and that the practical implementation examples we have given will be useful.

We would however like to point out that not all implementations have the same features and defaults, so before putting everything in production networks, we would strongly advise you to setup a test environment and do make sure that everything works as expected.

THE AUTHORS

Flavio Luciani

Flavio Luciani was born in Roma in 1981 and took his Master Degree in Computer Engineering from Roma 3 University in 2006. He has been working with Namex since 2008 and since then he has supervised the technical and infrastructural development of the exchange point, firstly as a member of the technical staff and then, starting from March 2020, serving as Chief Technology Officer.

Tiziano Tofoni

Tiziano Tofoni (a.k.a. Admiral Tofonoto) graduated in Electrical Engineering at the University of Padua (Italy) and holds a Master Degree in Mathematical Statistics from the Florida State University, Tallahassee, FL. He was a researcher at the Institute of Bioengineering and Dynamics of Systems of the Italian National Research Council (CNR) in Padua and Teaching Assistant at the Department of Statistics of the Florida State University. Subsequently, after a short experience in the Aerospace industry, he joined the staff of the Advanced School in Telecommunications "G. Reiss Romoli" (at that time part of Telecom Italia Group), where he has worked for over 20 years, in the field of Traffic Engineering, Advanced Technologies for IP Networks and Future Scenarios of Broadband Networks. He has held courses on Numerical Analysis, Probability and Queueing Theory at the Departments of Science and Engineering of the University of L'Aquila (Italy). He is a co- author of the book "Traffic Engineering in Telecommunication Networks", and is the author of "MPLS: Fundamentals and applications to IP networks", "BGP: from theory to practice" and "IS-IS: from theory to practice" (this last book is downloadable for free at http://blog.reissromoli.com). In 2010, along with other colleagues, he has (re-)founded the Reiss Romoli srl, heir to the great tradition of the Advanced School in Telecommunications G. Reiss Romoli. He is also a renowned blogger. You can reach his blog at http://blog.reissromoli.com.

Starting from November 1st 2019 he was appointed Chairman of the Board of Reiss Romoli srl.

This page intentionally left blank.