MASTER EN SCIENCES INFORMATIQUES

Contribution à l'évaluation d'une implémentation de BGPSec

Pacheco Cardena, Mateo

Award date:
2014

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2013–2014

Contribution à l'évaluation d'une implémentation de BGPSec

Mateo Pacheco

Promoteur : Laurent Schumacher (Signature pour approbation du dépôt - REE art. 40)

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Table des matières

Remerciements 4
Résumé 5
Table des acronymes utilisés 6
Introduction 7
1 Ressource Public Key Infrastructure 9
1.1 Hiérarchie d'un RPKI 9
1.2 Autorité de dépôt 11
1.3 Manifestes 13
1.4 Certificats 14
1.5 Révocation de certificats 16
1.6 Exemple commenté de RPKI 17
2 Route Origination Authorization 19
2.1 Description d'un ROA 19
2.2 Création et révocation d'un ROA 20
2.3 Validité d'une route 20
3 BGPSec 22
3.1 Limites du modèle RPKI-ROA 22
3.2 BGPSec 23
4 Présentation du testbed 25
4.1 La structure intra-routing 25
4.2 La structure inter-routing 27
4.3 La Structure du Testbed 29
4.4 Les limites du testbed 33
5 Scénarios de test 35
5.1 Statuts différents des routes validées et non validées 36
5.2 Update BGP à la suppression d'un ROA 36
5.3 Update BGP à l'expiration d'un certificat 36
6 Observations 37
6.1 Validité des routes malgré la mauvaise configuration des ROA . . . 38
6.2 Absence de réaction à l'expiration des certificats 42
6.3 Validité des routes malgré la suppression d'un ROA 42
6.4 Problèmes avec le validateur RIPE 43
Conclusions et perspectives 44

Remerciements

Je tiens à remercier mon promoteur, Laurent Schumacher, qui m'a aidé et conseillé tout au long du processus de mise en place du testbed ainsi que pour la rédaction de ce mémoire. Je tiens également à remercier Jérôme Maisonneuve sans l'aide de qui le testbed RPKI/BGPSec n'aurait pas pu être mis en place. Enfin, je tiens à remercier Ghislaine Lomba qui a bien voulu relire le présent document.

Résumé

Dans ce travail, nous allons, dans un premier temps, donner une vue d'ensemble d'une technique développée pour sécuriser le transport des données au sein d'un réseau : BGPSec. Ensuite, nous présenterons les testbed mis en place à la faculté d'informatique pour étudier ce protocole et les observations effectuées. BGPSec permet, par un système de certification, de créer des chemins sécurisés au sein du réseau. La mise en place de BGPSec nécessite une structure préexistante qu'on appelle *Ressource Public Key Infrastructure* (RPKI). Nous allons donc commencer ce travail en expliquant comment est organisé un RPKI et comment ses composants fonctionnent les uns avec les autres. Nous nous intéresserons ensuite aux routes et à leur origine. Nous verrons comment certifier que le point défini comme origine d'une route est effectivement l'origine de cette route. En d'autres termes, comment être sûr que rien n'a altéré cette donnée (ni par erreur, ni volontairement). Ensuite, nous expliquerons où BGPSec vient se greffer dans la structure qui a été mise en place jusque-là et comment son fonctionnement peut sécuriser une route. Dans la deuxième partie du travail, nous développerons l'implémentation mise en place pour tester BGPSec : le réseau et ses connexions intra et inter routing ainsi que la structure mise en place afin de permettre le fonctionnement de BGPSec. Nous présenterons, par la suite, les scénarios de tests proposés ainsi que les observations effectuées lors de ces tests au niveau du testbed. Enfin, nous terminerons en concluant et en donnant quelques pistes de travail possibles au vu des observations.

Table des acronymes utilisés

ANTD	Advanced Network Technologies Division
API	Application Programming Interface
AS	Autonomous System
BGP	Border Gateway Protocol
BRITE	BGPSec/RPKI Interoperability Test & Evaluation
CRL	Certification Revocation List
FAI	Fournisseurs d'Accès Internet
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigned Names and Numbers
IP	Internet Protocol
LIS	Local Internet Registry
MITM	Man in The Middle
NIR	National Internet Registry
NIST	National Institute of Standards and Technologies
OSPF	Open Shortest Path First
PKI	Public Key Infrastructure
RFC	Request for Comments
RIP	Routing Information Protocol
RIPE	Réseaux IP Européens
RIR	Regional Internet Registry
ROA	Route Origin Authorization
RPKI	Ressource Public Key Infrastructure
SIDR	Secure Inter Domain Routing
XML	Extensible Markup Language

Introduction

BGP est un protocole qui permet de définir des routes au sein d'un réseau permettant ainsi à un paquet de données d'aller d'un point d'origine à un point d'arrivée via le meilleur chemin. Pour ce faire, BGP fonctionne sur base de connections semi-permanentes qui définissent quels Systèmes Autonomes sont voisins les uns des autres. C'est ce qu'on appelle le routage interdomaine [14]. On nomme Système Autonome, un ensemble de noeuds du réseau (typiquement un ensemble de routeurs) ayant une politique de routage interne commune.

On peut, dès lors, facilement se rendre compte que BGP est un protocole fondamental d'un réseau et qu'un mauvais fonctionnement de ce protocole peut avoir de lourdes conséquences sur les transmissions de données. Or, aucun mécanisme de sécurisation n'a été implémenté au sein de BGP lors de sa conception comme le note S. Murphy [20, p. 1] "Il n'y a pas de mécanisme interne à BGP qui protège contre les attaques qui modifient, effacent, contrefont ou reproduisent les données alors que chacune [de ces attaques] a le potentiel de perturber l'ensemble du comportement de routage." En témoignent d'ailleurs un RFC intégralement dédié aux vulnérabilités de sécurité de BGP [20] ainsi que de nombreux articles qui en font part ou sont exclusivement consacrés à ces dernières. Citons, par exemple, l'article de Butler et al. [3] ou encore celui de Yanuzzi et al. [23].

De nombreuses solutions au manque de sécurité de BGP ont été proposées parmi lesquelles *Secure-BGP* (S-BGP) [13], *Secure Origin BGP* (soBGP) [22] ou encore Pretty Secure BGP (psBGP). La solution que nous avons décidé de présenter dans ce travail est appelée BGPSec. Elle se base sur une infrastructure qu'on nomme *Ressource Public Key Infrastructure* (RPKI), qui fournit une première sécurisation du réseau par un ensemble de certificats. Cette infrastructure crée un arbre hiérarchisé dont chaque branche est formée par une chaîne de certificats, chaque certificat étant signé par le certificat qui se trouve directement au-dessus de lui (en terme de graphe, un certificat fils est signé par son certificat père).

Le RPKI est la base d'un mécanisme qui garantit que le Système Autonome de départ de la route a bien le droit d'être l'origine de cette route, ce mécanisme est appelé le *Route Origination Authorization* (ROA). Mais, malgré le renforcement de la sécurité que fournissent le RPKI et le ROA lorsqu'ils travaillent de concert, ils ne sont que de peu d'utilité en ce qui concerne les tentatives d'attaques volontaires contre BGP et ne sont efficaces que lors de problèmes de configuration ayant un impact sur le routage du réseau. Comme nous le verrons, BGPSec est un prolongement du protocole BGP qui tente de pallier les faiblesses liées au ROA en utilisant un système d'encapsulation pour lier les différents Systèmes Autonomes et certifier, à chaque Système Autonome, que la route qui a été parcourue jusqu'à lui l'a été de façon sécurisée.

La section 1 de ce travail sera réservée à l'étude du RPKI. Nous commencerons par étudier la structure du RPKI puis les éléments qui le composent. Nous

verrons d'abord l'Autorité de Dépôt qui comprend les Points de Publication, où sont stockés notamment les certificats et ROA, et les Caches Locaux qui permettent à une entité de télécharger localement tous les certificats et ROA qui lui sont utiles. Ensuite, nous étudierons les Manifestes qui fournissent la liste des certificats et ROA contenus au sein d'un Point de Publication. Enfin, nous nous intéresserons aux différents types existants de certificats. Nous parlerons également de la façon dont on révoque un certificat. Enfin, pour conclure cette section, nous résumerons, à l'aide d'un exemple, le fonctionnement d'un RPKI. La section 2, elle, sera consacrée à la présentation du *Route Origination Authorization* (ROA) : sa description, comment il est créé et comment il est révoqué. Nous nous intéresserons également, à ce stade, à l'état de validité des routes BGP. La section 3 détaillera le mécanisme mis en place par les concepteurs de BGPSec pour sécuriser le routage BGP. Par la suite, dans la section 4, nous présenterons le testbed qui a été mis en place pour étudier le protocole BGPSec : sa structure intra-routing, sa structure inter-routing et la structure du RPKI et de BGPSec ainsi que les limites rencontrées lors de son implémentation et de son fonctionnement. La section 5, elle, s'occupera de présenter les scénarios de tests et la section 6, les réactions du testbed observées face aux différents scénarios de tests. Enfin, nous terminerons par quelques conclusions et perspectives sur base des observations effectuées.

1 Ressource Public Key Infrastructure

Un RPKI est une infrastructure mise en place pour supporter la sécurisation du routage interdomaine d'un réseau. Nous commencerons cette section en donnant un exemple de hiérarchie possible pour un RPKI. Cette hiérarchie suit la distribution des ressources IP telle qu'elle se fait sur le réseau Internet. Nous verrons ensuite où une entité qui fait partie du RPKI stocke un certificat ou un ROA, mais également comment une entité qui en a besoin les récupère. Nous détaillerons ensuite les types existants de certificats utilisés dans une infrastructure de type RPKI et comment ils sont révoqués. Enfin, nous terminerons par un exemple commenté montrant le fonctionnement général d'un RPKI.

1.1 Hiérarchie d'un RPKI

Selon l'agencement et l'architecture du réseau dans lequel on se trouve, beaucoup de solutions sont possibles pour structurer une Infrastructure à Clés Publiques (PKI). Le lecteur intéressé peut d'ailleurs en retrouver quelques-unes dans le RFC 4158 [5].

Dans le cas présent (un RPKI), le type d'Infrastructure à Clé Publique qui nous intéresse est développé comme support à la sécurisation du routage d'un réseau (en particulier Internet). Il est donc assez logique que la structure la plus adéquate pour la mise en place d'un RPKI suive la hiérarchie d'allocation de ressources IP déjà existante [16] [12]. Par ressources IP nous entendons les adresses IP (v4 et v6) ainsi que les numéros d'identification de Systèmes Autonomes.

La distribution de ressources IP est structurée comme suit : au sommet on trouve l'*Internet Assigned Number Authority* (IANA) qui est une branche de l'*Internet Corporation for Assigned Names and Numbers* (ICANN). L'IANA est responsable, entre autres, de la coordination des adresses IP et des numéros d'identification de Systèmes Autonomes au niveau mondial [1]. Elle est au sommet de l'arbre de distribution de ressources IP.

L'IANA distribue ces ressources à des *Regional Internet Registries* (RIR) qui, eux, sont chargés de leur gestion au sein d'une zone géographique donnée. Ils sont au nombre de cinq : AfriNIC pour l'Afrique, APNIC pour l'Asie-Pacifique, ARIN pour l'Amérique du Nord, LACNIC pour l'Amérique Latine et RIPE NCC pour l'Europe.

Ces RIR vont, à leur tour, distribuer ces ressources à des *National Internet Registries* (NIR) ou des *Local Internet Registries* (LIR) selon la zone géographique où se trouvent les ressources. Ce troisième niveau de prise en charge s'occupe, lui, de distribuer les ressources aux Fournisseurs d'Accès Internet (FAI).

La hiérarchie de l'allocation de ressources IP est donc une structure en arbre dont la racine est l'IANA et qui aboutit aux FAI. Les niveaux inférieurs aux FAI ne sont pas pris en compte dans cette hiérarchie. Cela est dû au fait que

la politique de sécurisation du routage internet se décide au niveau des FAI. L'utilisateur final n'a que très peu d'impact.

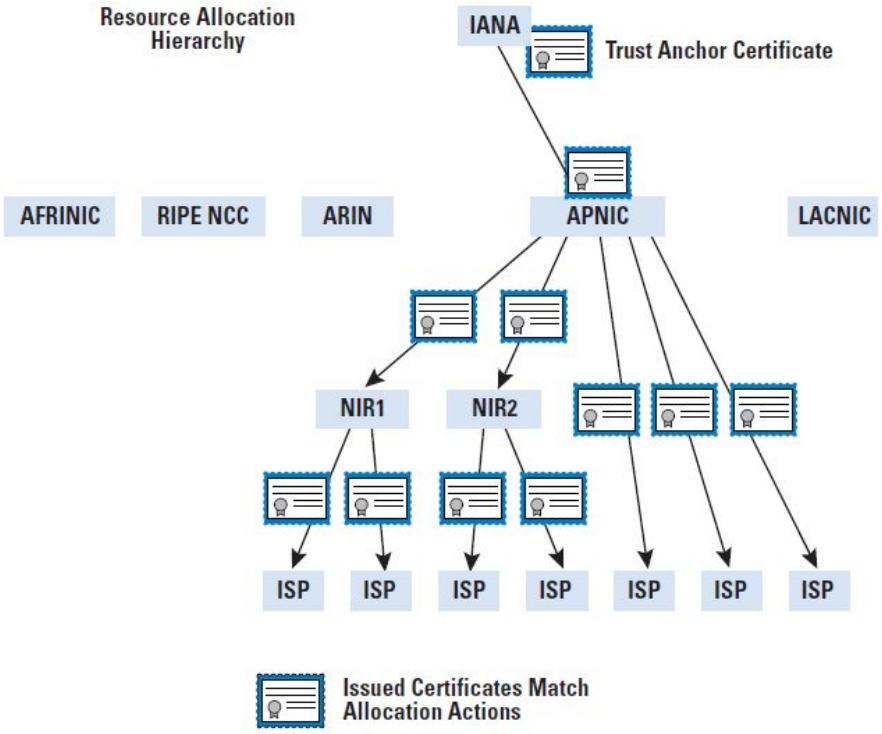

Figure 1. Hiérarchie d'un RPKI [12]

Comme le montre la Figure 1, la hiérarchie du RPKI reflète bien la structure pyramidale utilisée pour distribuer les adresses IP et numéros d'identification de Systèmes Autonomes.

Le RPKI est basé sur l'utilisation de Certificats de Ressources. Ces certificats attestent de l'allocation, par une autorité elle-même certifiée, de ressources IP à un sujet de niveau directement inférieur. L'autorité qui alloue les ressources IP est appelée Autorité de Certification car elle certifie que le sujet (c'est-à-dire celui qui a reçu les ressources) a bien autorité sur ces ressources. L'Autorité de Certification première, la racine de l'arbre, est appelée Point d'Ancrage. Elle signe elle-même son Certificat de Ressources. Les certificats utilisés par le RPKI

sont conformes au profil X.509 défini dans le RFC 5280 [4] ainsi qu'aux extensions nécessaires au transfert du droit d'utilisation d'adresses IP et de numéros de Systèmes Autonomes définis dans le RFC 3779 [19].

L'IANA, au sommet de la pyramide, est considéré comme un Point d'Ancrage, c'est-à-dire un point de confiance à partir duquel on peut vérifier la validité des certificats des niveaux inférieurs. Pour plus de détails sur les Points d'Ancrage, le lecteur est invité à se référer aux RFC 5914 [6] et 6490 [11].

Sous ce niveau 0 ou cette racine, on retrouve le niveau 1 qui, comme dans la hiérarchie d'allocation des ressources, contient les RIR qui possèdent tous des Certificats de Ressources. Ces certificats ont été validés par l'IANA en utilisant une signature digitale (c'est-à-dire en liant la clé publique du RIR et un nom unique donné au RIR au sein d'une signature faite à partir de la clé privée de l'IANA).

Les niveaux inférieurs fonctionnent de la même manière, chaque entité utilisant une signature digitale pour authentifier les certificats des entités du niveau qui lui est directement inférieur. Cette chaîne de confiance est ce qu'on appelle le Chemin de Certification c'est-à-dire le chemin à travers lequel on peut vérifier que le certificat d'une entité donnée est authentique [4]. Dès lors, lorsqu'une entité doit vérifier la validité d'un certificat donné, elle doit retrouver le Point d'Ancrage du Chemin de Certification du certificat en question et suivre la chaîne qui part de ce point pour arriver au certificat qu'elle cible.

Notons pour terminer que l'IANA et les RIR sont, au sein de la structure d'un RPKI, de très bons candidats au titre de Points d'Ancrage. Néanmoins, aucune politique n'est définie dans ce domaine et les entités qui utilisent un RPKI ont tout loisir de sélectionner les Points d'Ancrage de leur choix.

1.2 Autorité de dépôt

Les différentes entités qui forment la structure d'un RPKI éditent des objets signés [15]. Un objet signé est une structure de données particulière qui n'est pas un certificat, mais a été signé à l'aide d'un certificat par une Autorité de Certification.

Pour stocker ces objets signés, un RPKI utilise une structure de publication distribuée nommée l'Autorité de Dépôt [16] [12] [7]. Cette Autorité de Dépôt est composée de deux structures : les Points de Publication et les Caches Locaux.

Points de Publication

Chaque Autorité de Certification va, lorsqu'elle signe un objet, envoyer cet objet ainsi que le certificat à l'aide duquel elle a signé l'objet à un Point de Publication. L'Autorité de Certification a toute liberté dans le choix de l'endroit qui lui sert de Point de Publication. Une seule restriction s'applique : l'Autorité de Certification doit avoir un Point de Publication unique. Comme précisé dans

le RFC 6480 [16], l'ensemble des Points de Publication d'un RPKI fournit une information complète (c'est-à-dire exhaustive) des objets signés et certificats au sein du RPKI dont ils dépendent.

Nous nous intéressons, dans ce travail, à trois types d'objets signés qu'une Autorité de Certification peut publier :

- le manifeste : Il contient la liste des objets signés publiés au sein du point de publication dont il fait partie.
- la *Certification Revocation List* (CRL) : Cet objet contient la liste des certificats qui ont été révoqués par une Autorité de Certification. Chaque CRL est propre à une et une seule Autorité de Certification.
- le *Route Origin Authorization* (ROA) : il fournit l'autorisation explicite à un Système Autonome donné d'être à l'origine d'une route vers un ou plusieurs préfixes déterminés.

Une fois l'objet publié au sein du point de publication de l'Autorité de Certification qui l'a émis, il peut être consulté par toute Entité Utilisatrice qui en a besoin. On nomme Entité Utilisatrice, une entité qui utilise le RPKI pour s'assurer de l'authenticité d'un ensemble de certificats. Dans le cadre d'un RPKI, une entité utilisatrice est typiquement un FAI ou une LIR (voir 1.1).

Tous les objets présents au sein d'un point de publication sont signés, ce qui les protège de modifications illicites. En effet, si un objet modifié n'a pas été autorisé par l'Autorité de Certification dont il dépend, ce dernier n'est pas signé. Cela permet à l'Entité Utilisatrice de détecter que la modification n'est pas légitime.

Du point de vue de la gestion, chaque point de publication possède un manifeste (qui est un objet signé) [7] qui contient, pour chaque objet du point de publication, son nom ainsi qu'une valeur de hachage de son contenu. Cela permet d'assurer aux Entités Utilisatrices qu'elles ont pu consulter l'ensemble des objets et certificats contenus dans le Point de Publication déterminé. D'un point de vue sécurité, les Points de Publication ne sont pas conçus pour protéger la confidentialité des objets qu'ils contiennent. De ce fait, ce ne sont pas des endroits où mettre des données à caractère sensible.

Maintenance du cache local

En parallèle à ce système de dépôts, chaque Entité Utilisatrice va maintenir un cache local qui va lui permettre de télécharger les données qui l'intéressent et de les utiliser. En effet, pour utiliser un objet signé, une Entité Utilisatrice doit avoir une copie locale du Certificat d'Entité Finale inclus dans ledit objet comme nous le verrons par la suite (voir 1.4). Un Certificat d'Entité Finale est un type de certificat utilisé pour signer les objets.

Chaque Entité Utilisatrice doit donc régulièrement synchroniser les données de son cache avec celles du point de publication pour s'assurer qu'elles correspondent. La procédure est la suivante :

1. Tout d'abord, l'Entité Utilisatrice interroge l'Autorité de Dépôt et copie tous les certificats, manifestes et CRL édités au sein du RPKI.
2. L'Entité Utilisatrice vérifie, pour chaque Certificat d'Autorité, la signature correspondante dans le manifeste. Elle s'assure également que la date de la prochaine mise à jour du manifeste (indiquée dans le champ *nextUpdate*) n'a pas été atteinte ou dépassée.
3. Elle vérifie, via l'ensemble des manifestes du point de publication, que les certificats et la CRL présents correspondent bien aux valeurs de hachage contenues dans le manifeste où ils apparaissent (voir 1.3). Mais également qu'aucun objet apparaissant dans les manifestes ne manque ou n'a été ajouté dans le point de publication.
4. Enfin, elle doit valider chaque Certificat d'Entité Finale en construisant et en vérifiant son chemin de validation. Comme expliqué dans la hiérarchie du RPKI (voir 1.1), cela se fait sur base d'un chemin "topdown" : elle collecte d'abord les informations liées aux Points d'Ancrage choisis et télécharge les Certificats d'Autorité de ces Points d'Ancrage. A partir de cet ensemble de certificats, elle va pouvoir aller chercher tous les certificats valides qui ont été émis par le Point d'Ancrage. Cette opération est répétée sur les certificats ainsi téléchargés et ce jusqu'à arriver aux Certificats d'Entité Finale.

Pour terminer, notons que, comme précisé dans le RFC 6480 [16], puisque les Entités Utilisatrices synchronisent régulièrement leur cache avec les points de publication, il est intéressant (en termes de performance) de n'effectuer ces opérations que sur les objets mis à jour depuis la dernière synchronisation.

1.3 Manifestes

Un manifeste est un objet signé qui contient la liste des objets signés (sauf le manifeste lui-même) qui ont été émis par une Autorité de Certification au sein du point de publication qu'elle utilise [7]. Le manifeste étant lui-même un objet signé, il contient un Certificat d'Entité Finale unique (voir 1.6), signé par l'Autorité de Certification qui utilise le point de publication dans lequel il se trouve. De plus, le champ *nextUpdate* [2] permet de vérifier s'il a été altéré ou pas. Un manifeste sera considéré comme ayant expiré si la date courante est plus récente que la date qui figure dans ce champ.

Lorsqu'une Entité Utilisatrice veut construire son cache local ou le synchroniser avec un point de publication [16], elle vérifie que chaque objet qu'elle trouve dans le point de publication apparaît bien dans le manifeste et inversement. Ensuite, l'Entité Utilisatrice vérifie que chaque valeur de hachage associée à un

objet dans le manifeste correspond à la valeur obtenue en hachant l'objet du point de publication [2].

L'utilisation de cet objet signé permet de limiter les risques de mauvaise synchronisation suite à l'effacement d'un objet ou à son remplacement par une version plus ancienne, par exemple [16]. Ce mécanisme est très important puisque les points de publication qui forment le système de dépôts du RPKI, sont, eux, non sécurisés.

Comme expliqué dans le RFC 6486 [2], la structure du RPKI ne tolère pas qu'un point de publication soit vide. Dès lors, lorsqu'un point de publication est créé, il contient au moins un manifeste dans lequel il y a au moins une entrée : la CRL du point de publication (voir 1.5).

1.4 Certificats

Une infrastructure RPKI utilise deux types de Certificats de Ressources : les Certificats d'Autorité et les Certificats d'Entité Finale. Ces deux types de certificats sont conformes au profil défini par le groupe *Secure Inter Domain Routing* (SIDR) pour ce type de certificats [10].

Certificats d'Autorité

Le rôle des Certificats d'Autorité est d'attester la légitimité d'un détenteur de ressources IP, c'est-à-dire d'adresses IP ou de numéros de Systèmes Autonomes, à détenir ces ressources. Cette certification se fait en liant une clé publique aux ressources IP allouées via une signature digitale [16].

Il existe trois classes de Certificats d'Autorité qui correspondent à trois situations différentes [4] :

- Le *cross-certificate* : où l'éditeur du certificat et l'Entité qui reçoit les ressources (que nous appellerons le sujet) sont deux entités différentes mais toutes deux sont des Autorités de Certification. Ici, trois cas de figure sont possibles :
  1. Le sujet est une Autorité de Certification qui a le droit de sous-allouer ses ressources IP. Il va, dès lors, recevoir un Certificat d'Autorité et sera capable, lui-même, d'émettre ce type de certificats. Au sein de la hiérarchie du RPKI (voir 1.1) cela se produit par exemple lorsque un RIR émet un Certificat d'Autorité pour un NIR.
  2. Le sujet est une Autorité de Certification qui n'a pas le droit de sous-allouer ses ressources IP. Dès lors, il recevra un Certificat d'Entité Finale.
  3. Le sujet est une Autorité de Certification qui ne peut pas sous-allouer ses ressources mais doit pouvoir émettre des objets (et donc les signer). Il recevra un Certificat d'Autorité.

- Le *self-issued certificate* : qui correspond au fait que l'éditeur du certificat et le sujet sont la même entité. Ce genre de cas survient s'il y a changement dans la politique de routage du RPKI ou lorsqu'une Autorité de Certification effectue une procédure de changement de clés, c'est-à-dire lorsqu'elle met à jour son couple clé privée/clé publique (voir RFC 6489 [9]).

- Le *self-signed certificate* : qui est un *self-issued certificate* dont la signature peut être vérifiée à l'aide de la clé publique liée au certificat lui-même. Ils sont nécessaires pour commencer les chemins de certification et sont donc utilisés par les Points d'Ancrage.

Ainsi, une Autorité de Certification, qui détient un ensemble de ressources et veut en allouer une partie à une entité qui dépend d'elle, va utiliser son Certificat d'Autorité pour ce faire. Elle va émettre un certificat, où sont mentionnées les ressources qu'elle alloue et une clé publique, et va lier ces deux informations en signant le certificat à l'aide de sa clé privée. Pour vérifier le certificat il ne reste plus alors qu'à suivre le chemin à partir du Point d'Ancrage, la clé publique d'une Autorité de Certification décryptant les certificats qu'elle a émis.

Bien sûr, chaque certificat atteste de l'allocation d'un ensemble de ressources à un détenteur déterminé. Ainsi, si un détenteur possède des ressources de plusieurs sources différentes, il aura plusieurs Certificats d'Autorité.

Pour conclure ce point sur les Certificats d'Autorité, il nous faut souligner deux choses :

- Bien que l'Autorité de Certification vérifie l'identité de l'Entité à laquelle elle alloue ses ressources, son rôle n'est pas de certifier cette identité. De ce fait, le nom utilisé dans le certificat ne décrit en rien l'Entité pour laquelle il a été émis. Le Certificat d'Autorité est donc un outil d'autorisation mais pas d'authentification.
- La réédition d'un Certificat d'Autorité, par exemple pour changer le couple clé publique/clé privée, n'implique pas de rééditer les certificats qui ont été édités par lui. En effet, ces derniers ont un champ appelé *Authority Information Access* (AIA) qui contient l'adresse de l'endroit où est publié le Certificat d'Autorité. Si la clé publique est changée alors il faut rééditer un certificat pour l'Autorité de Certification dont le couple clé publique/clé privée a changé et signer à nouveau, à l'aide du nouveau couple de clés, la chaîne de certificats partant du Certificat d'Autorité réédité. Il n'est donc pas nécessaire de rééditer l'entièreté des certificats composant la chaîne de certification partant de l'Autorité de Certification. Si on ne change pas le couple clé publique/clé privée mais qu'on ne fait que recréer une nouvelle clé privée sur base de la clé publique existante, la seule contrainte d'une réédition de Certificat d'Autorité est de conserver le même endroit de publication.

Certificats d'Entité Finale
Les Certificats d'Entité Finale sont émis par un détenteur de ressources. Il est donc nécessaire pour une entité déterminée de posséder un Certificat d'Autorité pour pouvoir émettre un certificat d'Entité Finale [16].

Ces Certificats d'Entité Finale permettent à l'entité dont il sont issus de déléguer l'autorité qu'elle possède sur un sous-ensemble de ses ressources. En particulier, ce type de certificat est utilisé pour vérifier la validité d'objets signés tels que les *Route Origin Authorisation* (ROA) ou les Manifestes. Le certificat se trouve dans l'extension *certificates* de l'objet qu'il vérifie [15] ce qui permet de ne pas devoir envoyer l'objet et son certificat en deux parties distinctes.

La procédure de création d'un objet signé à l'aide d'un Certificat d'Entité Finale est la suivante :

- On sélectionne un sous-ensemble des ressources IP détenues par l'Autorité de Certification qui émet le Certificat d'Entité Finale. Ce sous-ensemble devient le sujet du certificat.
- L'Autorité de Certification génère un certificat d'Entité Finale qui reprend la liste des ressources sélectionnées. Cette liste confirme le sous-ensemble de ressources IP constituant le sujet de l'objet.
- L'Autorité de Certification spécifie les dates pour lesquelles le Certificat d'Entité Finale est valide.
- Enfin, l'autorisation (c'est-à-dire l'objet signé et son Certificat d'Entité Finale) est publiée dans le point de publication de l'Entité Finale.

Au niveau de la gestion de leurs clés, les certificats d'Entité Finale ont deux particularités intéressantes. D'une part, la clé privée qui forme un couple avec la clé publique contenue dans le Certificat d'Entité Finale est à usage unique. D'autre part, un objet n'est signé que par une et une seule clé privée. Cela crée donc une correspondance 1-1 entre un objet et son certificat qui fournit deux avantages :

1. Pour révoquer un objet signé, une Autorité de Certification doit simplement révoquer le Certificat d'Entité Finale contenu dans cet objet (puisque ce certificat est unique).
2. Puisqu'une clé privée de certificat d'Entité Finale n'est jamais utilisée pour signer deux objets différents, on peut la détruire après utilisation. Cela permet d'alléger la gestion des clés.

1.5 Révocation de certificats

Pour révoquer un certificat, les Autorités de Certification vont utiliser un type particulier d'objet signé, la *Certification Revocation List* (CRL) [4]. Chaque Autorité de Certification édite régulièrement une CRL unique qui contient tous les certificats révoqués, et qui est disponible dans son point de publication. La

fréquence à laquelle une CRL est éditée dépend du cycle de mise à jour de l'Autorité de Certification dont elle dépend (ce cycle est propre à chaque Autorité de Certification). Rappelons que, puisque pour révoquer un objet signé il suffit de révoquer son certificat d'Entité Finale, le CRL n'a besoin de référencer que des certificats.

Les raisons de révocation d'un certificat sont multiples : renouvellement des clés, réduction de l'ensemble des ressources allouées au bénéficiaire du certificat ou encore fin de l'allocation des ressources [12]. Le processus de révocation d'un certificat donné est le suivant [4] :

1. Lors de la mise à jour suivant l'obsolescence du certificat, l'Autorité de Certification ajoute une nouvelle entrée dans la CRL correspondant au certificat. Chaque certificat révoqué est identifié dans la CRL par son numéro de série.
2. Une fois la mise à jour faite, la nouvelle CRL est uploadée dans le point de publication de l'Autorité de Certification.
3. Lorsqu'une Entité Utilisatrice utilise un certificat, elle vérifie la signature et la validité du certificat (c'est-à-dire les dates pour lesquelles le certificat est valide). Elle récupère également la dernière version de la CRL du point de publication et vérifie que le numéro de série du certificat n'y apparaît pas.

L'avantage de révoquer un certificat de cette manière [4] est que la CRL peut être distribuée de la même manière que les certificats c'est-à-dire via des communications inter serveurs non sécurisées. En contrepartie, l'inconvénient est que la granularité du temps de révocation dépend de la fréquence de mise à jour de son point de publication par l'Autorité de Certification. Cela retarde d'autant la notification du certificat comme révoqué.

1.6 Exemple commenté de RPKI

La Figure 2 montre une représentation logique d'un exemple de RPKI. Elle est basée sur la Figure 6 du RFC 4158 [5, p. 15] et la Figure 1 du draft sidr-bgpsec-pki-profiles-03 [21, p. 3].

La partie gauche du schéma représente le Chemin de Certification qui permet de vérifier l'authenticité d'un certificat ou d'un objet signé. Dans notre exemple, le RPKI est formé d'un Point d'Ancrage (root), de deux Autorités de Certification (A et B) et d'une Entité Finale (EE). Le Chemin de Certification se traduit, sur la Figure, par les arêtes doubles qui relient les entités du RPKI entre elles. Chaque entité qui participe au RPKI possède un Point de Publication. C'est là qu'elle stocke les objets et certificats qu'elle signe.

Sur la Figure, les objets et certificats édités dans un Point de Publication sont représentés dans un cadre en pointillés. La notation "Certificat X(Y)" signifie que le Certificat est émis par l'Autorité de Certification Y pour certifier l'Autorité de Certification X. L'arête en pointillé, qui part du certificat, rejoint

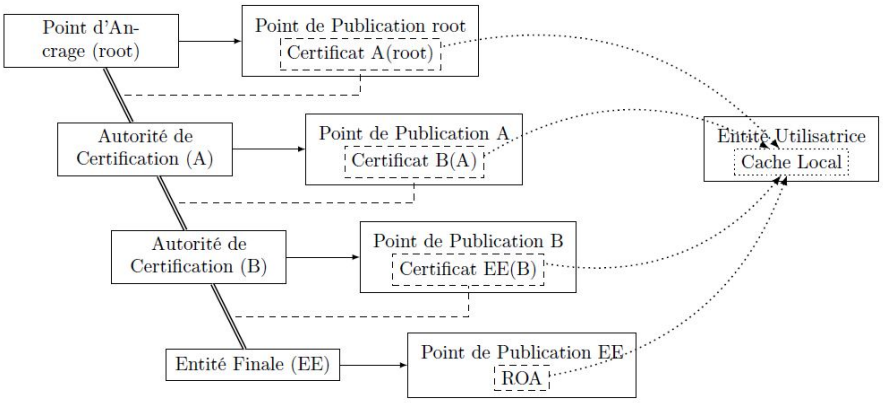

Figure 2. Schéma logique d'un RPKI

la portion du Chemin de Certification sur lequel il a autorité.

La droite de la Figure montre une Entité Utilisatrice ainsi que son Cache Local. L'Entité télécharge les objets et certificats contenus dans les différents Points de Publication comme en témoignent les arcs formés de points qui arrivent à son Cache Local. Cela va lui permettre de vérifier que le ROA qui se trouve dans le Point de Publication de l'Entité Finale, est bien valide.

Pour vérifier l'authenticité d'un objet qu'elle a téléchargé, l'Entité Utilisatrice va parcourir le Chemin de Certification depuis le Point d'Ancrage jusqu'à l'objet ciblé. Dans notre exemple (la Figure 2), l'Entité Utilisatrice va vérifier la validité du ROA émis par l'Entité Finale (EE) :

- L'Entité part du Point d'Ancrage et vérifie que le certificat émis par celui-ci est authentique en décryptant le "Certificat A(root)" à l'aide de la clé publique du Point d'Ancrage. Ce faisant, elle récupère la clé publique de l'Autorité de Certification A qui se trouvait dans le certificat. Elle peut, dès lors, vérifier si la clé publique annoncée par l'Autorité de Certification A est bien la même que celle qui se trouve dans le "Certificat A(root)".
- Si c'est le cas, l'Entité Utilisatrice procède de la même manière entre les Autorités de Certification A et B puis entre l'Autorité de Certification B et l'Entité Finale.
- Enfin, l'Entité Utilisatrice décrypte le Certificat d'Entité Finale (contenu dans le ROA) à l'aide de la clé publique de l'Entité Finale (EE). Elle peut, dès lors, vérifier si les ressources annoncées par le ROA sont identiques à celles annoncées dans le Certificat d'Entité Finale. Si tel est le cas, le ROA est valide.

2 Route Origination Authorization

Le *Route Origin Authorization* (ROA) est un objet signé qui confirme la légitimité de l'origine d'une route. Il est signé à l'aide d'un Certificat d'Entité Finale. Cette section décrit le ROA et ce qu'il contient, explique comment créer un ROA et le révoquer. Enfin, elle se termine en faisant le point sur l'état de validité d'une route dans le cadre de l'utilisation de ROA.

2.1 Description d'un ROA

Comme indiqué dans le RFC 6480 [16], les informations fournies par le RPKI ne sont pas suffisantes pour prendre les décisions liées au routage. En particulier, le protocole BGP considère que si un Système Autonome veut accéder à un ensemble de préfixes particuliers, c'est qu'il en a obtenu le droit. Il nous faut donc un outil capable de vérifier qu'effectivement le détenteur des préfixes IP auxquels un Système Autonome veut accéder a donné son accord pour cet accès. Cet outil c'est le *Route Origin Authorization* (ROA), décrit en détail dans le RFC 6482 [17].

Un ROA comprend deux parties distinctes [16] :
- Le contenu du ROA lui-même, composé notamment d'un numéro d'identification de Système Autonome (le Système Autonome à l'origine de la route) ainsi que de la liste des préfixes d'adresses IP (les préfixes qui constituent la destination de la route).
- un certificat d'Entité Finale qui contient normalement les mêmes préfixes d'adresses IP que ceux contenus dans le ROA.

Pour que le Certificat d'Entité Finale soit valide, il faut que la liste des préfixes qu'il contient soit identique à celle du ROA qu'il certifie et inversement. Notons qu'une autre utilisation des ROA est possible dans un cadre de vérification. Un FAI pourrait très bien utiliser les ROA valides pour les préfixes de son réseau comme données d'entrée, dans le but de construire des filtres pour les routes utilisées par ses routeurs BGP. Pour plus d'informations, voir le RFC 6483 [8].

Notons également que chaque ROA possède un seul numéro de Système Autonome [16]. Donc, le nombre de ROA vers un préfixe donné dépend du nombre de Systèmes Autonomes autorisés à être le point d'origine d'une route vers ce préfixe. Autrement dit, si plusieurs Systèmes Autonomes peuvent être le point de départ d'une route vers un préfixe (scénario de multi-homing), ils ont chacun leur propre ROA qui les y autorise.

Comme tout objet signé du RPKI, les ROA sont distribués à partir des points de publication des différentes Autorités de Certification [16] puisque c'est là que sont concentrés tous les outils nécessaires à leur vérification (c'est-à-dire les cer-

tificats et les CRL).

Pour terminer, signalons l'existence d'un ROA particulier : le ROA qui contient le numéro d'identification de Système Autonome 0. En effet, ce dernier signifie qu'aucun Système Autonome ne peut être à l'origine d'une route BGP vers les préfixes qui figurent sur ce ROA.

2.2 Création et révocation d'un ROA

Pour créer un ROA, un détenteur de ressources doit avoir le statut d'Autorité de Certification. En effet, il doit posséder un certificat de ressources lui permettant d'allouer le ou les préfixes pour lesquels il s'apprête à émettre un ROA [16], c'est-à-dire un Certificat d'Autorité.

La procédure pour émettre un ROA est la suivante :

1. L'Autorité de Certification doit créer un Certificat d'Entité Finale qui contient les préfixes autorisés à être la destination de la route.
2. Ensuite, elle construit la payload du ROA c'est-à-dire la partie de l'objet composé des informations fondamentales à transmettre. Cela inclut les préfixes IP du Certificat d'Entité Finale et le numéro d'identification du Système Autonome à l'origine de la route.
3. Elle signe le ROA avec la clé privée correspondante à la clé publique contenue dans le Certificat d'Entité Finale.
4. Elle uploade le ROA et le Certificat d'Entité Finale dans le point de publication qu'elle utilise.

Pour révoquer un ROA, comme pour révoquer tout objet, il suffit de révoquer son certificat d'Entité Finale. L'Autorité de Certification doit donc :

- créer une CRL.
- uploader la CRL dans le point de publication qu'elle utilise.
- enlever le ROA révoqué ainsi que le Certificat d'Entité Finale correspondant du point de publication.

Malgré tout, il faut faire attention lorsqu'on révoque un ROA. En effet, cette action peut poser un problème si aucun autre ROA valide n'existe pour la liste des préfixes IP mentionnés dans le ROA révoqué. Si tel est le cas, cela peut entraîner l'impossibilité d'envoyer des paquets de données vers ces préfixes IP. Il est donc préférable, pour le détenteur des ressources, de vérifier avant de révoquer un ROA, qu'il existe un autre ROA valide pour ces préfixes.

2.3 Validité d'une route

La validité d'une route dépend de la correspondance entre les préfixes IP et le numéro de Système Autonome contenu dans le ROA, et ceux contenus dans

le Certificat d'Entité Finale qui signe le ROA [8]. Dans le cadre d'une adoption partielle des ROA, c'est-à-dire une situation où seulement un sous-ensemble des préfixes d'un RPKI sont associés à un ou des ROA, la procédure de validation d'une route est la suivante :

1. on sélectionne les ROA dont les préfixes correspondent aux préfixes annoncés par la route. Cet ensemble de ROA est appelé "ROA candidats".
2. Si l'ensemble de "ROA candidats" est vide, alors l'état de validité de la route est considéré comme "Inconnu".
3. Si l'ensemble de ROA candidats est non vide et qu'un des ROA a le même numéro de Système Autonome et le même préfixe IP que la route, alors la route est considérée comme "valide".
4. Sinon, la route est considérée comme "non valide".

Table 1. Etat de validité des routes

numéro d'AS / préfixe	**correspond au ROA**	**différent du ROA**
correspond au ROA	valide	non valide
plus spécifique que ROA	non valide	non valide
non décrit entièrement	inconnu	inconnu

La Table 1 nous montre un résumé des valeurs de validité que peut prendre une route :

- Correspondance : si les détails de la route (préfixes IP et numéro de Système Autonome) correspondent à ceux d'un ROA valide, la route est considérée comme "valide".
- Non Correspondance : si le numéro de Système Autonome ou les préfixes annoncés par la route ne correspondent à ceux d'aucun ROA, alors la Route est "non valide" (typiquement c'est le cas si les préfixes annoncés par la route sont plus spécifiques que ceux des ROA valides).
- Correspondance Partielle : si la route a un préfixe qui n'est décrit entière -ment par aucun ROA et que ce préfixe n'est pas plus spécifique que ceux de ROA valides, alors son état de validité est défini comme "Inconnu".

Bien sûr, l'interprétation de ces états de validité reste une question de politique locale propre aux entités en charge du routage sur le réseau. Notons toutefois que, dans le cas d'une adoption partielle des ROA, l'état de validité "Inconnu" ne devrait pas être adopté par une politique de routage comme argument de rejet d'une route. En effet, un réseau qui n'utilise que partiellement les ROA aura forcément des préfixes non couverts par un ROA. Dès lors, rejeter une route vers ces préfixes (pour ce seul motif) empêchera tout transfert de données vers ces préfixes.

Pour terminer, signalons que la validité d'une route n'est valable que pendant un laps de temps déterminé. Au-delà de cette limite, la validité de la route incriminée doit être vérifiée.

3 BGPSec

Dans les sections précédentes de ce travail, nous avons présenté le RPKI ainsi que le ROA et montré comment ils travaillent de concert pour fournir un premier mécanisme de sécurisation du routage interdomaine. Nous allons maintenant, au sein de cette section, faire part des limites de ce modèle RPKI-ROA puis montrer de quelle manière BGPSec permet de pallier ces faiblesses.

3.1 Limites du modèle RPKI-ROA

BGP utilise des connexions semi-permanentes pour définir les routes existantes au travers d'un réseau. Ces routes sont mises à jour au travers de messages de mise à jour nommés messages d'UPDATE. Ainsi, lorsqu'une nouvelle route est créée, chaque routeur BGP propage l'information à ses voisins via un message d'UPDATE pour que ces derniers aient connaissance de la nouvelle route [14] [3]. Pour déterminer les routes qui seront utilisées au sein du réseau, les routeurs BGP doivent conserver une trace des Systèmes Autonomes traversés par la route pour arriver jusqu'à eux. Et si un routeur voit que la route est déjà passée par lui auparavant, il refuse que la route ne repasse par lui. En effet, si la route repassait deux fois par le même routeur, cela voudrait dire qu'elle fait une boucle.

L'attribut qui conserve la trace, au sein du message d'UPDATE, du chemin parcouru s'appelle l'AS_PATH. Il est composé des numéros de série des Systèmes Autonomes par lesquels le paquet de données est déjà passé. L'AS_PATH se construit au fur et à mesure de la propagation de la route, chaque système autonome ajoutant son numéro lorsque le paquet de données est propagé au Système Autonome suivant. Donc, si un routeur frontière (c'est-à-dire un routeur se trouvant à la frontière entre un Système Autonome et ses voisins) voit que l'AS_PATH contient déjà son numéro de Système Autonome, il ne permettra pas que la route passe une nouvelle fois par lui. Le gros problème de BGP, et donc de la mécanique de routage qu'il met en place, est qu'il n'est pas sécurisé contre les éventuelles erreurs (au sein du réseau) provenant de mauvaises configurations ni, a fortiori, contre les attaques malveillantes qui ont pour but de déterminer l'usage d'une route.

Comme expliqué précédemment, le ROA fournit un premier mécanisme de sécurisation de BGP (en particulier de l'AS_PATH) en validant l'origine d'une route. Malgré tout, si le procédé valide la propagation d'une route à partir d'un Système Autonome, il ne garantit pas que le Système Autonome d'origine soit bien le Système Autonome où la route a commencé. Une personne malveillante

pourrait très bien créer une route qui apparaîtrait comme venant du Système Autonome valide alors que ce n'est pas le cas. Un ROA est donc très utile pour les accidents ou les erreurs de configuration mais n'a que peu d'utilité en ce qui concerne les attaques malveillantes.

La question qui se pose est donc de savoir : est-ce que la séquence de numéros de Système Autonome correspond bien à la route qui a été empruntée par le datagramme ? Ou, en d'autres termes, est-ce que quelqu'un est venu s'intercaler à un moment et à un endroit déterminé et a changé les données contenues dans l'AS_PATH.

3.2 BGPSec

Ce que propose BGPSec pour compenser les faiblesses du RPKI est de mettre en place une nouvelle structure au niveau du routeur via un nouvel élément de certification : le certificat de routeur [18] [12].

Le but de ce certificat est de certifier au routeur local la route qui s'étend depuis le Système Autonome d'origine jusqu'à lui. Cette certification se fait à l'aide d'un nouvel attribut nommé "BGPSec_Path_Signature" qui vient s'ajouter au message d'UPDATE déjà existant. Ce nouvel attribut est une séquence de couples signatures/valeurs de hachage de clé publique.

Chaque signature peut être décryptée par la clé publique qui l'accompagne et est composée :
- des signatures précédentes ou du préfixe de destination si le message d'UPDATE se trouve à son point d'origine.
- du numéro de Système Autonome local.
- du numéro du prochain Système Autonome à atteindre pour poursuivre la route.

A chaque propagation de la route, le couple signature et valeur de hachage est ajouté à la valeur de l'attribut BGPSec_Path_ Attribute.

La Figure 3 montre un exemple de sécurisation du routage BGP via BGPSec. Comme on peut le voir, la route mise en place part d'un Système Autonome (AS) numéroté 1 et aboutit à l'AS 3.

A l'AS 1 (l'AS d'origine), le routeur frontière récupère le préfixe d'adresses IP de départ, le numéro d'AS local, le numéro d'AS de destination et la clé publique. Il signe le tout à l'aide d'une clé privée de manière à créer une signature. Il accole à cette signature la valeur de hachage de la clé publique (qui se trouve dans la signature) et met le tout dans l'attribut BGPSec_Path_Attribute. Puis il envoie le message d'UPDATE BGP à l'AS 2.

Lors du passage de l'AS 2 à l'AS 3 : l'AS 2 récupère la signature de l'AS 1 et y accole son numéro d'identification, le numéro d'identification de l'AS suivant

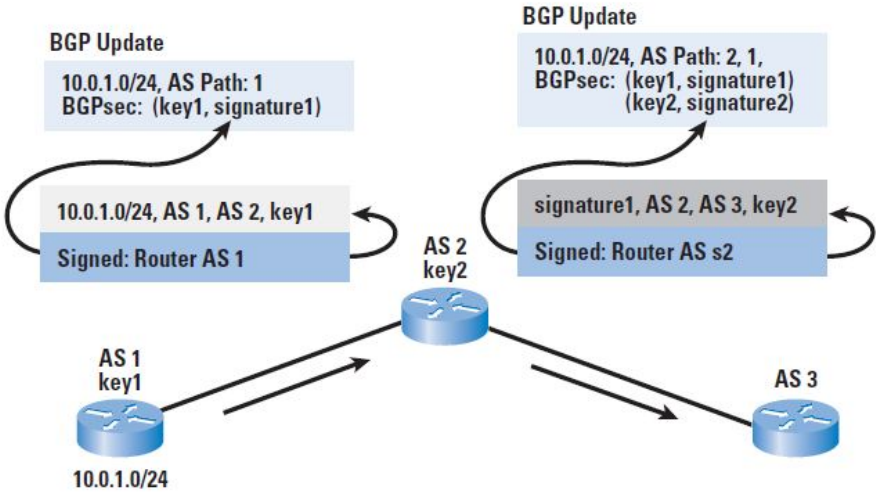

Figure 3. Exemple de sécurisation du routage par BGPSec [12]

et une clé publique. Il signe le tout, y accole la valeur de hachage de la clé publique, et l'envoie au routeur suivant de la route. Ce processus se répète de la même façon jusqu'à ce que le parcours entier de la route ait été mis à jour.

Avec ce système d'authentification des routes, le routeur BGP est, dès lors, à même de vérifier :

- si le chemin parcouru par la route correspond bien au chemin contenu dans l'attribut AS_PATH (attribut du protocole BGP).
- si le Système Autonome d'origine de la route est autorisé à déclarer une route vers les préfixes mentionnés. Cette vérification se fait par l'intermédiaire du ROA (voir section 2).

Enfin, signalons que, dans un modèle d'adoption partielle de BGPSec, il est nécessaire de négocier son utilisation lors de l'ouverture de la session BGP. Cette négociation doit être faite au niveau de chaque routeur pour chaque famille d'adresses (IPv4 et IPv6). La négociation doit également tenir compte du fait que le routeur BGP peut envoyer des mises à jour BGPSec, les recevoir ou les deux. Une fois la négociation terminée, les messages d'UPDATE BGP et BGPSec peuvent être envoyés pour mettre à jour la route.

Notons que BGPSec n'est pas destiné à faire du tunneling à travers les parties du réseau qui ne supportent pas BGPSec. Ainsi, si au sein de cette route un routeur BGP se rend compte que le routeur auquel il doit envoyer son message ne supporte pas BGPSec, il doit enlever toutes les mises à jour BGP-

Sec_Path_Signatures de ce qu'il envoie à ce routeur.

4 Présentation du testbed

Le testbed utilisé pour étudier le protocole BGPSec (en collaboration avec une implémentation de RPKI) comprend trois systèmes autonomes qui communiquent entre eux au travers d'un réseau *full mesh*.

Le testbed d'étude de BGPSec a été mis en place à l'aide de 6 machines :
- une machine travaillant sous Debian 6 et fonctionnant en tant que serveur de certificat. Cette machine contient l'implémentation RPKI et forme ainsi le début de l'arbre de certification.
- deux machines travaillant sous openSUSE 11.2 et fonctionnant comme pseudo-routeur BGPSec (c'est-à-dire des routeurs supportant la validation des routes au travers du protocole BGPSec).
- trois routeurs Cisco 7200 Series qui jouent le rôle de routeurs "standards" (c'est-à-dire des routeurs qui ne supportent pas la validation des routes au travers du protocole BGPSec).

La figure 4 montre le réseau mis en place en tant que testbed pour l'étude du protocole BGPSec. Le serveur RPKI correspond à la machine tournant sous Debian 6 et qui forme le début de l'arbre de certification. Les routeurs 1, 2 et 5 sont les trois routeurs Cisco 7200 Series et les routeurs 3 et 4, les pseudo-routeurs BGPSec tournant sous openSUSE 11.2. Nous reviendrons, plus loin, sur les différents éléments qui composent ce réseau ainsi que sur leurs interconnexions.

4.1 La structure intra-routing

La structure intra-routing du testbed comprend trois systèmes autonomes respectivement nommés "AS1", "AS2" et "AS3". La figure 5 montre la découpe logique de la structure du réseau local implémenté pour construire le testbed.

AS1 et AS2

Les systèmes autonomes "AS1" et "AS2" sont tous deux composés de deux machines :
- une machine (ordinateur) fonctionnant sous openSUSE 11.2 qui joue le rôle de pseudo-routeur BGPSec et est utilisé comme Entité Utilisatrice. Cette machine récupère les informations BGP qui permettent de mettre à jour sa table de routage. Ces informations sont accompagnées des statuts de validité des routes. Le pseudo-routeur peut ainsi déterminer quelle route est valide, quelle route est non valide et quelle route a un statut de validité non déterminé.

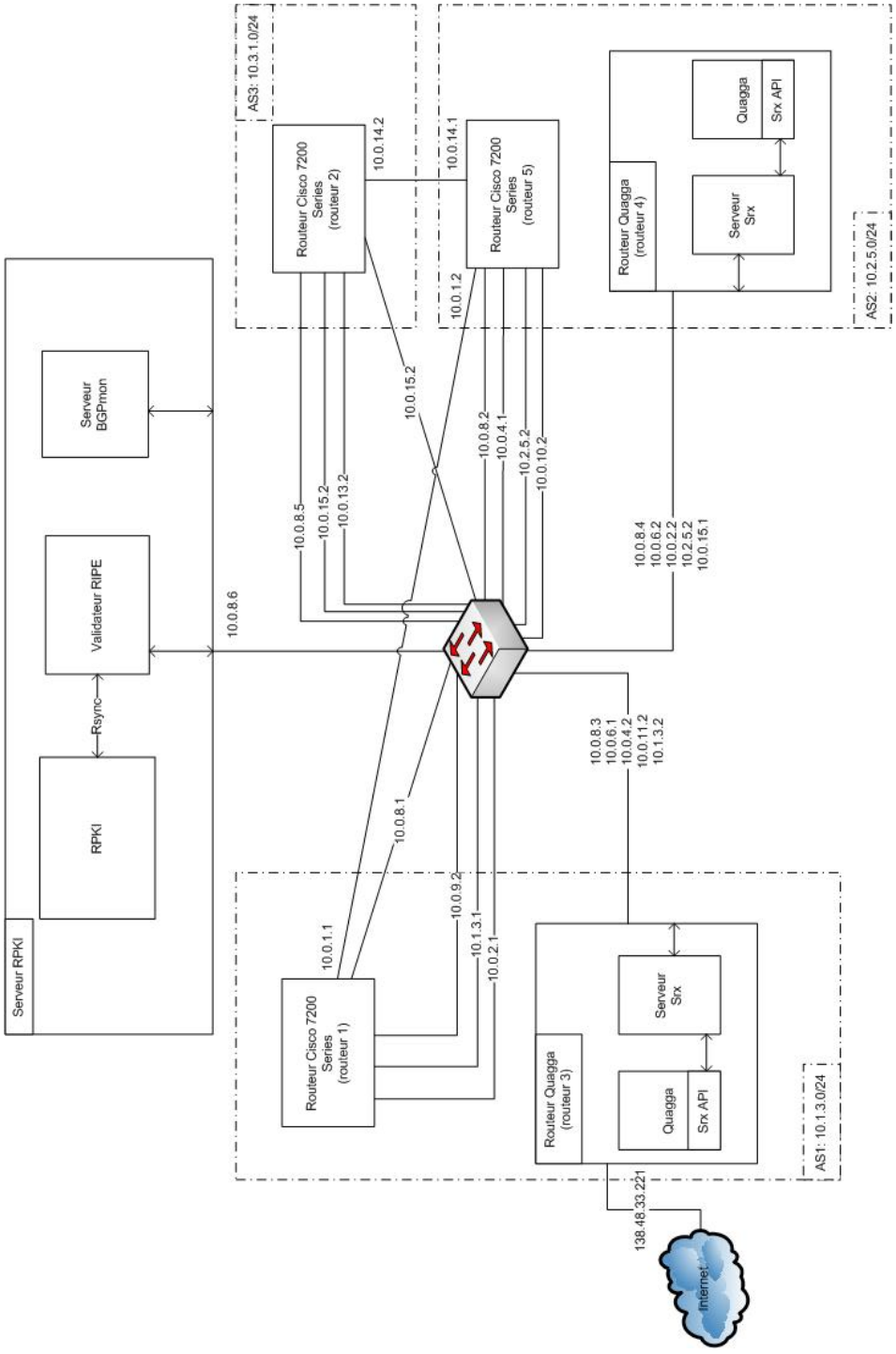

Figure 4. Schéma général du testbed d'étude BGPSec

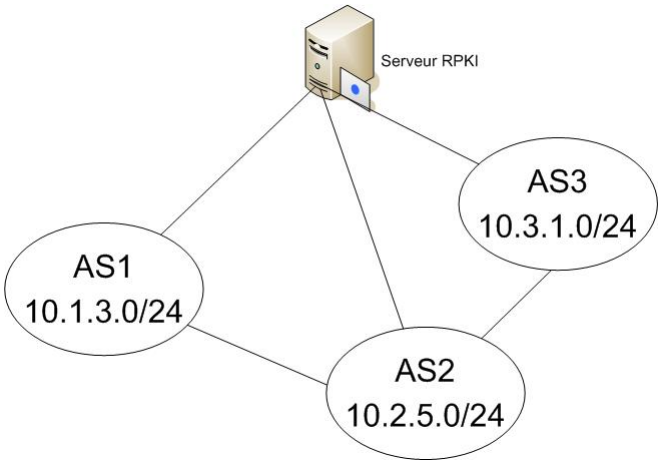

Figure 5. Découpe logique du réseau

– la seconde machine utilisée au sein des systèmes autonomes "AS1" et "AS2" est un routeur Cisco 7200 Séries qui, lui, ne supporte pas le protocole BGPSec. Ces routeurs ont été mis en place au sein du testbed pour permettre une circulation intra-domaine des paquets.

Comme le montre la figure 5, "AS1" est responsable des adresses IP définies par le sous-réseau 10.1.3.0/24. A l'intérieur de ce sous-réseau :
– le pseudo-routeur BGPSec communique via l'interface 10.1.3.2.
– le routeur Cisco 7200 Series communique via l'interface 10.1.3.1.

"AS2" est responsable des adresses IP définies par le sous-réseau 10.2.5.0/24. A l'intérieur de ce sous-réseau :
– le pseudo-routeur BGPSec communique via l'interface 10.2.5.2.
– le routeur Cisco 7200 Series communique via l'interface 10.2.5.1.

AS3
Le système autonome "AS3", lui est uniquement composé d'un routeur Cisco 7200 Series. Il a été mis en place pour permettre une meilleure circulation inter-domaine des paquets au sein du réseau.
"AS3" est responsable des adresses IP définies par le sous-réseau 10.3.1.0/24. A l'intérieur de ce sous-réseau, il possède l'adresse 10.3.1.1.

4.2 La structure inter-routing

Au sein du réseau implémenté pour le testbed, plusieurs sous-réseaux ont été définis pour le routage inter-domaine des paquets. Ces sous-réseaux créent :

- un réseau de communication de type full mesh entre les différents systèmes autonomes présents.
- un réseau de communication en étoile entre le serveur RPKI et les autres dispositifs hardware du testbed. Ces communications permettent les mises à jour des statuts de validité des ROA.

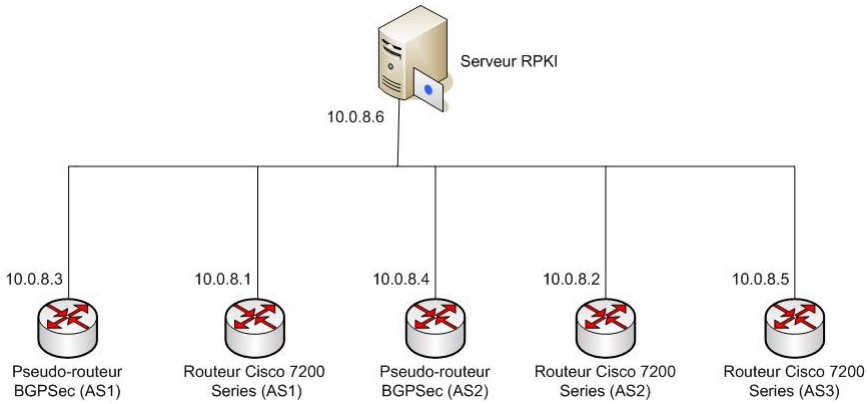

Figure 6. Schéma logique des liens de communications entre le serveur RPKI et les composants du réseau

Comme le montre la figure 6, les communications entre le serveur RPKI et les autres machines du réseau se font via le sous-réseau 10.0.8.0/24 :

- le serveur RPKI utilise l'interface 10.0.8.6.
- le routeur Cisco 7200 Series de "AS1" utilise l'interface 10.0.8.1.
- le routeur Cisco 7200 Series de "AS2" utilise l'interface 10.0.8.2.
- le pseudo-routeur BGPSec de "AS1" utilise l'interface 10.0.8.3.
- le pseudo-routeur BGPSec de "AS2" utilise l'interface 10.0.8.4.
- le routeur Cisco 7200 Series de "AS3" utilise l'interface 10.0.8.5.

Comme le montre la figure 7, les communications entre les machines du réseau full mesh implémenté utilisent les sous-réseaux suivants :

- le sous-réseau 10.0.6.0/24 relie le pseudo-routeur BGPSec de "AS1" (qui utilise l'interface 10.0.6.1) au pseudo-routeur BGPSec de "AS2" (qui utilise l'interface 10.0.6.2).
- le sous-réseau 10.0.4.0/24 relie le pseudo-routeur de "AS1" (qui utilise l'interface 10.0.4.2) au routeur Cisco 7200 Series de "AS2" (qui utilise l'interface 10.0.4.1).
- le sous-réseau 10.0.1.0/24 relie le routeur Cisco 7200 Series de "AS1" (qui utilise l'interface 10.0.1.1) au routeur Cisco 7200 Series de "AS2" (qui utilise l'interface 10.0.1.2).

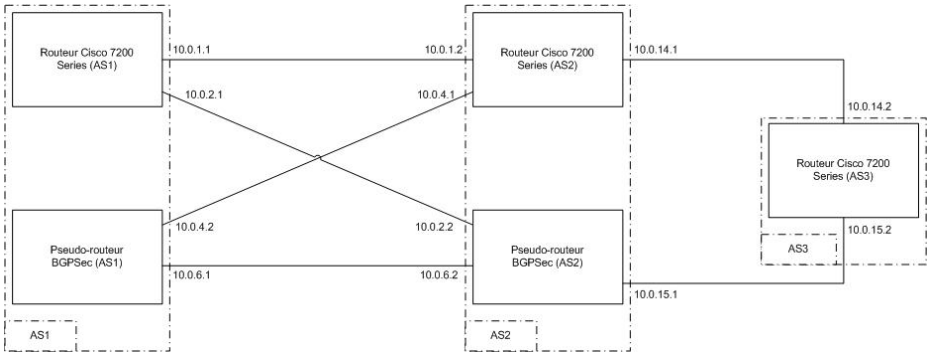

Figure 7. Schéma logique du réseau full mesh du testbed BGPSec

- le sous-réseau 10.0.2.0/24 relie le routeur Cisco 7200 Series de "AS1" (qui utilise l'interface 10.0.2.1) au routeur pseudo-routeur BGPSec de "AS2" (qui utilise l'interface 10.0.2.2).
- le sous-réseau 10.0.14.0/24 relie le routeur Cisco 7200 Series de "AS1" (qui utilise l'interface 10.0.14.1) au routeur Cisco 7200 Series de "AS3" (qui utilise l'interface 10.0.14.2).
- le sous-réseau 10.0.15.0/24 relie le pseudo-routeur de "AS2" (qui utilise l'interface 10.0.15.1) au routeur Cisco 7200 Series de "AS3" (qui utilise l'interface 10.0.15.2).

La figure 8 montre la découpe physique du réseau local configuré pour contenir le testbed telle qu'elle apparaît au sein du laboratoire de la faculté d'informatique. Comme le montre cette figure, toutes les communications, au sein du réseau du testbed, passent par le switch général du laboratoire de la faculté d'informatique qui est en charge de diriger les paquets vers leur destination.

4.3 La Structure du Testbed

Le serveur RPKI joue le rôle de Point d'Ancrage ainsi que d'Autorité de Certification pour le réseau local mis en place et forme ainsi le début du chemin de certification.

L'implémentation du RPKI utilisée sur cette machine est une implémentation opensource téléchargeable sur le site "http ://rpki.net". Comme le montre la figure 9, le paramétrage du RPKI sur le serveur RPKI comprend les entités suivantes :

- Un noeud racine appelé "trust_anchor" jouant le rôle de point d'Ancrage. C'est ce noeud racine qui crée et édite le certificat de Point d'Ancrage à partir duquel tous les autres certificats vont être créés.
- Une Autorité de Certification appelée "rir1", fils du noeud racine, qui est responsable de l'allocation des ressources IP définies par le sous-réseau

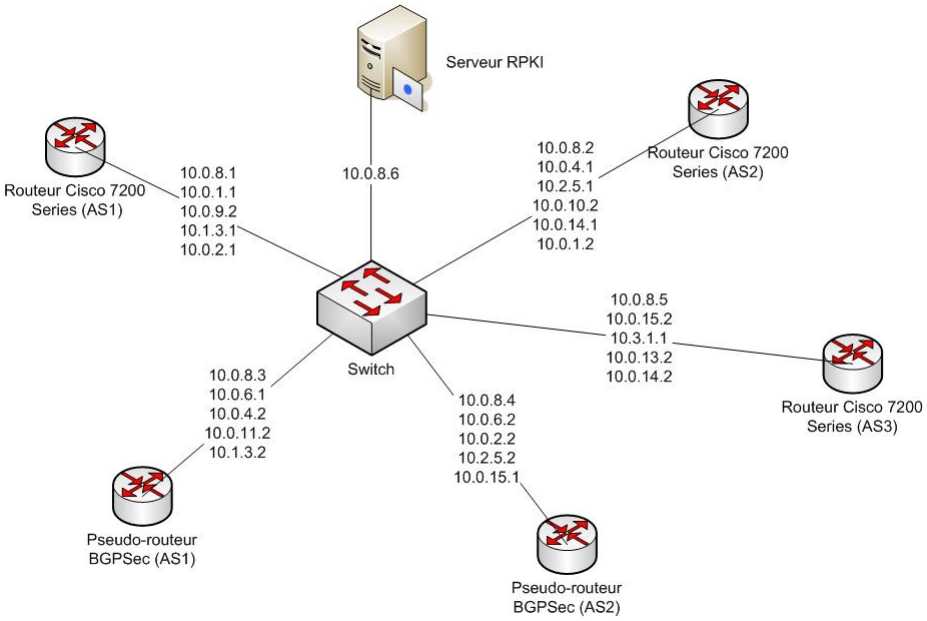

Figure 8. Découpe physique du réseau

10.1.0.0/16. On trouve au sein de ce sous-réseau, le Système Autonome "AS1" à qui rir1 a délégué le sous-réseau 10.1.3.0/24 et pour qui il a émis un ROA.

– Une Autorité de Certification appelée "rir2", fils du noeud racine, qui est responsable de l'allocation des ressources IP définies par le sous-réseau 10.2.0.0/15. On trouve au sein de ce sous-réseau les Systèmes Autonomes "AS2" et "AS3" à qui "rir2" a délégué respectivement les sous-réseaux 10.2.5.0/24 et 10.3.1.0/24.

Certificat racine

Le certificat racine a été créé automatiquement à l'aide de l'implémentation openssl fournie par l'implémentation RPKI installée sur le serveur. C'est sur base de ce certificat racine que le RPKI crée les certificats d'autorité des deux autorités de certification (rir1 et rir2). Ces certificats certifient les sous-réseaux utilisés pour le routage des paquets BGP au sein du réseau local mis en place.

Sur base des certificats d'autorité, le RPKI crée les ROA utilisés pour certifier les points de départ des routes définies au sein du réseau local mis en place.

Une fois l'arbre de certification mis en place, le RPKI a besoin de pouvoir transmettre les données vers les Entités Utilisatrices. Le programme utilisé pour effectuer cette tâche est le logiciel RIPE NCC Validator 2.11. Ce logiciel permet aux Entités Utilisatrices (les pseudos-routeurs BGPSec) de récupérer les données

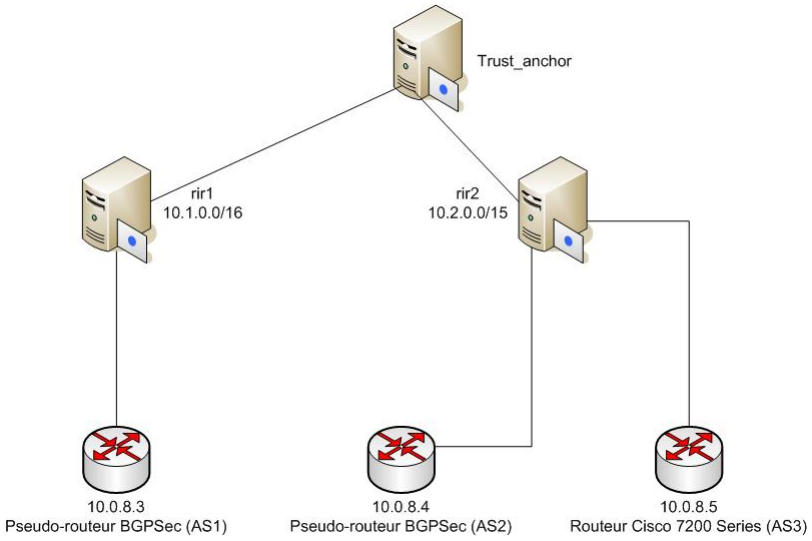

Figure 9. Schéma des chemins de certification créés au sein testbed

présentes au sein du Point de Publication du RPKI et de les conserver au niveau de leur cache local. Ce sont ces données qui permettent, en suivant le chemin de certification, de valider le départ d'une route.

Lorsqu'ils ont récupéré les informations transmises par le validateur RIPE, les pseudos-routeurs BGPSec doivent vérifier si les routes contenues dans leur table de routage sont valides, non valides ou si leur validité est non déterminée. Pour ce faire, deux logiciels sont installés sur les pseudos-routeurs BGPSec, Quagga 0.99.16 et Srx 0.2.0 :

- Quagga est un logiciel permettant de router des paquets sur des réseaux TCP/IP. Il permet le routage interne et externe via l'implémentation des protocoles RIP, OSPF et BGP.
- Srx est un composant additionnel qui gère la validation de l'origine ainsi que la validation du chemin (permettant ainsi à un routeur d'utiliser BGP-Sec). Ce module fonctionne à la manière d'un serveur proxy qui s'interface avec le logiciel du routeur (en l'occurence Quagga) via une API.

A titre d'exemple, la figure 10 montre la configuration du routeur Quagga de "AS1" ainsi que la configuration du serveur proxy qui lui est associé.

Enfin, pour permettre cette étude d'un RPKI et du protocole BGPSec associé, un outil de monitoring appelé "BGPmon" a été mis en place. Cet outil récupère les paquets BGP émis par un ou plusieurs routeurs et les renvoie sous forme d'un flux XML. Il est alors possible de récupérer ces paquets en clair pour voir l'évolution chronologique de la situation.

```
Current configuration:
!
hostname bgpd
password zebra
log file /usr/local/sbin/bgpd.log
log stdout
log syslog
!
debug bgp as4
debug bgp events
debug bgp keepalives
debug bgp updates
debug bgp zebra
!
router bgp 1
 bgp router-id 10.1.3.2
 bgp log-neighbor-changes
 network 10.1.3.0/24
 neighbor 10.0.4.1 remote-as 2
 neighbor 10.0.6.2 remote-as 2
 neighbor 10.0.11.1 remote-as 10
 !SRx Configuration Settigns
 srx display
 srx connect localhost 17900
 srx evaluation bgpsec
 srx keep-window 900
!
line vty
!
end
bgpd(config-router)# show srx-config
SRx configuration settings:
  server.......: localhost
  port.........: 17900
  proxy-id.....: 1
  keep-window..: 900
  evaluation...: bgpsec (prefix-origin and path processing)
  policy.......: ignore-undefined
  connected....: true
bgpd(config-router)#
```

Figure 10. Configuration du pseudo routeur Quagga ainsi que de son proxy Srx

BGPmon comprend également un module appelé "BGPmon Archiver" qui, une fois configuré, permet de récupérer les paquets et de les classer dans des fichiers horodatés. D'autre part, afin de faciliter la recherche d'éléments en particulier dans les données envoyées par BGPmon, un parseur a été mis en place. Ce dernier permet de parser les informations envoyées dans le stream par BGPmon et d'envoyer ces données dans une table MySQL.

4.4 Les limites du testbed

Nous nous sommes heurtés à deux limites avec le testbed mis en place pour étudier BGPSec. La première est la tentative de validation de la structure du testbed à l'aide du testeur RPKI/BGPSec BRITE ; cette tentative n'a pas fonctionné. La seconde est l'impossibilité au vu des logiciels utilisables pour BGPSec de mettre en place un réseau complètement virtualisé via le simulateur GNS3.

Validation BRITE

BRITE (BGPSec/RPKI Interoperability Test & Evaluation) est un framework web-based permettant de tester diverses implémentations de sécurisation du protocole BGP via le RPKI et BGPSec. Ce framework a été développé par l'ANTD (Advanced Network Technologies Division) qui est une division de la NIST (National Institute of Standards and Technologies).
Dans le but d'observer si notre implémentation du RPKI et BGPSec réagissait correctement, c'est-à-dire atteignait le résultat escompté par les scénarios de tests proposés par BRITE, nous avons tenté d'exécuter plusieurs scénarios de tests de la plate-forme BRITE. Il n'a néanmoins jamais été possible d'obtenir les résultats des tests passés puisque, une fois lancés, les scénarios de test tentent de se connecter au testbed sans jamais y arriver.

Comme le montre la figure 11, le problème qui se pose au niveau de l'utilisation de la plate-forme BRITE pour tester notre implémentation de RPKI/BGPSec est double :

- tout d'abord, comme le montre le cadre "Configuration Instructions", il semble que la plate-forme BRITE attende certaines configurations très précises tel que des numéros de systèmes autonomes définis (ici le routeur BGP doit porter le numéro d'AS 3000). De plus, certaines options (comme les vérifications multihop eBGP) doivent être désactivées.
- le second problème posé par la plate-forme BRITE est que le serveur RPKI de l'implémentation à tester doit être accessible puisqu'il doit être connecté à BRITE (dans l'exemple présenté sur la figure 11 à l'adresse 129.6.100.10 :50000). Or, le testbed mis en place à la faculté d'informatique ne permettait pas d'accéder directement au serveur RPKI depuis l'extérieur. En effet, l'unique accès extérieur vers le testbed se situe au niveau du pseudo routeur BGPSec de "AS1" (l'adresse WAN étant 138.48.33.221).

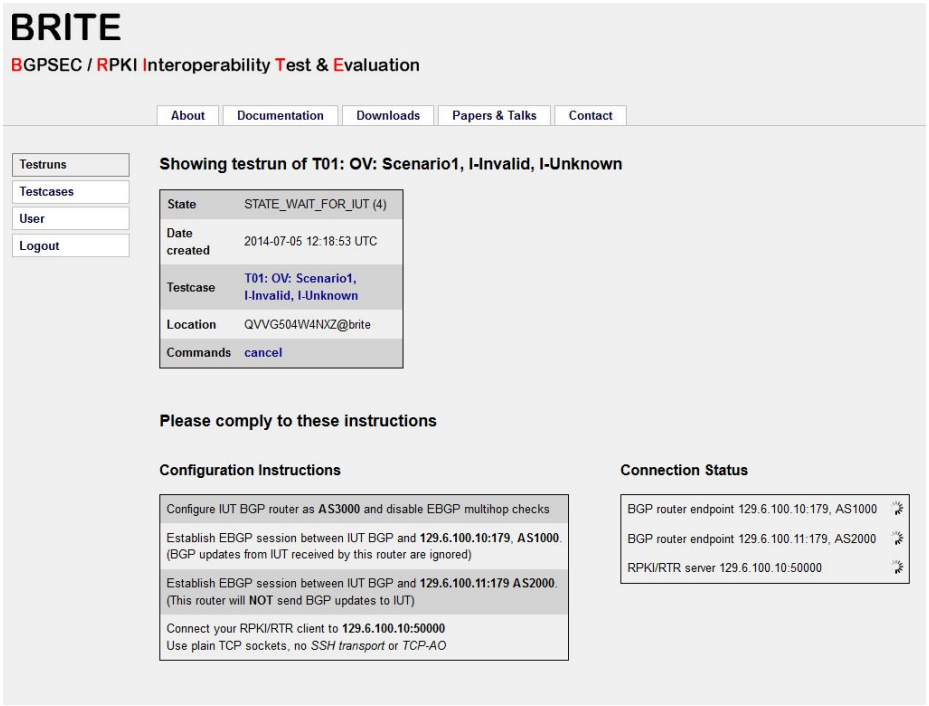

Figure 11. Capture d'écran de la plate-forme BRITE lors de l'exécution d'un test

Sachant que la plate-forme BRITE fonctionne sur le principe d'une black-box, il est difficile d'affirmer que les problèmes évoqués ci-dessus soient les seules causes de l'impossibilité de valider l'implémentation du testbed. Malgré tout, ces problèmes semblent tout de même être à l'origine de la non compatibilité de BRITE et du testbed BGPSec.

GNS3

GNS3 est un logiciel permettant de créer des simulations de réseau (que ce soit au niveau des technologies utilisées comme au niveau de la structure du réseau). Ce logiciel, qui supporte une série de technologies (routeur, switch,...) permet de définir une topologie réseau. Pour que cette topologie fonctionne, néanmoins, il est nécessaire d'installer pour chaque dispositif une version de son système d'exploitation.
Le problème posé par GNS3 pour créer une topologie BGPSec est que le support des validateurs RPKI par le système d'exploitation Cisco IOS n'est assuré qu'à partir de la version 15.2, et cette version de Cisco IOS nécessite l'utilisation, au minimum, de routeurs Cisco série 7600. Or, GNS3 ne supporte pas les routeurs Cisco de la série 7600. Cela nous a donc empêché de créer une topologie qui nous aurait permis de tester une autre implémentation et voir si cette dernière fonctionnait correctement.

5 Scénarios de test

Le but poursuivi lors de la mise en place du testbed était d'étudier les prises de risques associées aux préfixes annoncés. Il était donc prévu, dans un premier temps, d'identifier le fonctionnement du protocole BGPSec lors d'un certain nombre de cas limites afin de déterminer si certaines de ces situations amenaient à un comportement erroné de validation de routes.

D'autre part, il était également prévu de faire une étude des performances de BGPSec. L'étude devait porter sur les éventuelles périodes de latence lors desquelles les objets du RPKI ont été mis à jour sans que les routeurs n'aient encore eu le temps de mettre à jour la validation des routes concernées. Le but de cette seconde partie était d'avoir une idée du temps nécessaire à un routeur pour mettre à jour sa table de routage avec les nouvelles informations liées aux objets du RPKI et à la validation par le protocole BGPSec. Cette deuxième étude avait pour but de définir si le fonctionnement de BGPSec permettait un niveau de performance suffisant ou bien si la complexité des objets utilisés amenait un risque d'attaque. En d'autres termes, il nous fallait vérifier si les temps de latence nécessaires aux routeurs pour mettre à jour leur table de routage sur base des nouvelles informations des objets du RPKI ainsi que de BGPSec auraient permis une attaque sur une route non mise à jour.

5.1 Statuts différents des routes validées et non validées

Ce scénario de test avait pour but de déterminer si l'implémentation BGP-Sec du testbed gérait correctement les différents statuts de validation des routes.

Les résultats attendus sont représentés au sein du tableau suivant :

Table 2. Matrice des états de validité supportés par Quagga/Srx

ROA \ chemin	**valide**	**non valide**	**non déterminé**
valide	valide	non valide	non déterminé
inconnu	non valide	non valide	non valide
non valide	non valide	non valide	non valide
non déterminé	non déterminé	non valide	non déterminé

Comme le montre le tableau 2, Quagga/Srx utilise 4 valeurs pour valider le statut d'un ROA et 3 valeurs pour déterminer le statut du chemin qui part du ROA et arrive à destination :

- le statut d'une route est "valide" si son point d'origine est certifié par un ROA valide et que son chemin est correctement validé par le protocole BGPSec.
- le statut d'une route est "non déterminé" si le ROA validant son point d'origine est lui-même "non déterminé" mais que le chemin est valide, si au contraire le ROA est "valide" mais le chemin "non déterminé" ou si ROA et chemin sont "non déterminés". Signalons que ce statut est utilisé par Quagga/Srx comme statut temporaire lorsque la validation du ROA, du chemin ou des deux n'a pas encore eu lieu.
- enfin, tous les cas qui ne sont pas repris ci-dessus amènent à des routes "non valides".

5.2 Update BGP à la suppression d'un ROA

Ce scénario de test avait pour but de vérifier que lorsqu'une route est invalidée suite à la suppression d'un ROA, Quagga/Srx met correctement à jour les données de sa table de routage.

Comme le montre le tableau 2, le résultat attendu pour la route concernée est la mise à jour de son statut de validité depuis un statut "valide" vers un statut "non valide".

5.3 Update BGP à l'expiration d'un certificat

Ce scénario de test avait pour but de vérifier que lorsqu'une route est invalidée suite à la l'expiration d'un certificat, Quagga/Srx met correctement à jour

les données de sa table routage.

Comme le montre le tableau 2, le résultat attendu pour la route concernée est la mise à jour de son statut de validité depuis un statut "valide" vers un statut "non valide".

6 Observations

Depuis la mise en place du testbed, nous avions déjà pu observer un manque total de réaction du testbed aux différents tests réalisés. Les observations consignées alors, ont donc, dans un premier temps, cherché à savoir dans quelle partie du testbed se situait la cause de ce manque de réaction.

Lors de la phase finale de la rédaction du présent document (plus précisément le 14 août 2014), nous avons remarqué un problème avec la configuration des ROA tels qu'il avaient été configurés dès la mise en place du testbed. Les observations ont donc dû être revues et adaptées.

En effet, comme précisé dans la sous-section 2.1, un ROA est un objet signé qui autorise un Système Autonome à être à l'origine d'une route vers une destination donnée. Dès lors, le contenu d'un ROA comprend :

- un numéro de Système Autonome (le Système Autonome d'origine de la route)
- une liste de préfixes (les préfixes qui constituent la destination de la route)

Or, si nous analysons les ROA qui ont été définis au sein du testbed, nous remarquons que le préfixe contenu dans ces ROA (qui doit normalement définir la destination de la route) correspond, à chaque fois, au préfixe de sous-réseau du Système Autonome d'origine de la route contenu dans ce même ROA.

Autrement dit, nous avons créé trois ROA qui, chacun, certifient un Système Autonome comme origine valide d'une route à destination de lui-même :

- "AS1" (dont le sous-réseau est 10.1.3.0/24) est un point d'origine valide pour une route à destination du préfixe 10.1.3.0/24.
- "AS2" (dont le sous-réseau est 10.2.5.0/24) est un point d'origine valide pour une route à destination du préfixe 10.2.5.0/24.
- "AS3" (dont le sous-réseau est 10.3.1.0/24) est un point d'origine valide pour une route à destination du préfixe 10.3.1.0/24.

Ce problème n'a été relevé que très tardivement car le testbed affichant les routes attendues comme valides, nous n'avons pas pensé que la configuration des ROA pouvait être liée au problème de non réaction du testbed.

Cette observation soulève donc une question : Comment le testbed a-t-il pu valider des routes dont le point d'origine n'était pas certifié par un ROA ?

Dans les pages qui suivent, nous allons donc faire part des observations que nous avions effectuées (avant la découverte de cette erreur de configuration) à la

lumière de ce nouvel élément. Il ne s'agit, ici, bien sûr, que d'une approche tout à fait théorique puisque aucun nouveau test n'a pu être effectué sur le tesbed depuis la découverte de ce problème (faute de temps).

6.1 Validité des routes malgré la mauvaise configuration des ROA

A la lumière de l'erreur de configuration des ROA qui ont été mis en place au sein du testbed, on peut se demander pourquoi le système a considéré comme valides les routes BGP annoncées au sein des tables de routage des pseudo-routeurs BGPSec.

En effet, la configuration des ROA telle qu'elle a été réalisée permet de valider les Système Autonome comme point d'origine d'une route vers eux-même. Deux cas de figure sont donc possibles pour les routes :

- la route a une longueur 0.
- la route a une longueur N et forme une boucle.

Or, aucun de ces deux cas ne peut expliquer la validation par Quagga et Srx des routes.

En effet, si la route a une longueur 0, il est évident qu'elle ne peut être considérée comme "valide" puisqu'elle ne passe par aucun autre Système Autonome que celui de son point de départ. Ce type de route est d'ailleurs visible dans la table de routage du pseudo-routeur BGPSec (voir figure 16) en tant que route considérée (selon la matrice des états de validité supportés par Quagga/Srx, voir table 2) comme "non déterminée", le ROA ayant un statut "non déterminé" et le chemin BGPSec ayant également un statut "non déterminé. Ces routes apparaissent également avec un poids de 32768 pour éviter qu'elles ne soient prises en compte lors du routage des paquets.

En ce qui concerne le cas d'une route de longueur N et formant une boucle, aucune route de ce type n'apparaît au sein des tables de routage des pseudo-routeurs BGPSec. Et cela est tout à fait logique. En effet, d'une part, comme précisé au point 3.1, BGP possède un attribut nommé AS_PATH qui répertorie les numéros de Système Autonome par lesquels la route est déjà passée. C'est cet attribut qui est utilisé pour vérifier que la route n'effectue pas de boucle (un Système Autonome refusant à une route de passer deux fois par lui). Il est donc impossible qu'une route formant une boucle pour revenir à son point de départ soit définie au sein d'une table de routage. On peut en conclure que les trois ROA définis au sein du testbed ne peuvent valider aucune route puisque aucune route d'un point A vers lui-même ne peut être définie par BGP.

En ce qui concerne la validation du chemin par BGPSec, le protocole BGPSec utilise un attribut nommé BGPSec_Path_Signatures qui comprend un couple (valeur de hachage de clé publique/signature) pour chaque routeur par lequel est passée la route. Chaque routeur de la route définie récupère le couple du routeur précédent, y ajoute ses propres informations et ajoute le couple ainsi formé à la suite des signatures présentes dans l'attribut BGPSec_Path_Signatures. Ce n'est

qu'arrivé à destination que le processus de décryptage des différentes signatures est réalisé, ce qui permet à l'entité de destination de vérifier que les paquets n'ont pas dévié de la route annoncée. Ce processus et le ROA qui certifie le point d'origine de la route travaillent de concert ; il faut que l'endroit où se trouve le paquet lorsqu'il est arrivé à destination et la destination définie au sein du ROA de cette route soient identiques. Il est donc théoriquement impossible que des routes soient considérées comme "valides" si l'endroit d'arrivée et la destination définie dans le ROA ne correspondent pas.

En conclusion, trois mécanismes auraient dû empêcher les routes d'être considérées comme valides :

- BGP qui ne permet pas à une route d'effectuer une boucle.
- le ROA qui ne certifie pas les Systèmes Autonomes comme origine valide pour les préfixes de destination des routes notées comme "valides" au sein des tables de routage des pseudo-routeurs BGPSec.
- BGPSec qui, une fois le paquet arrivé à destination, a besoin d'un ROA avec le préfixe de destination correct pour valider le chemin.

Le flux de données Pour mieux comprendre la façon dont les mises à jour des tables de routage des pseudos-routeurs BGPSec sont réalisées, nous présentons ici le flux de données qui correspond à ces mises à jour (la situation présentée ici est postérieure à la suppression du ROA de "AS1", voir point 6.3).

Comme en témoignent les figures 12 ainsi que 13, les ROA définis au sein du RPKI sont les suivants :

- un ROA validant "AS2" (sous-réseau 10.2.5.0/24) comme point d'origine valide d'une route.
- un ROA validant "AS3" (sous-réseau 10.3.1.0/24) comme point d'origine valide d'une route.

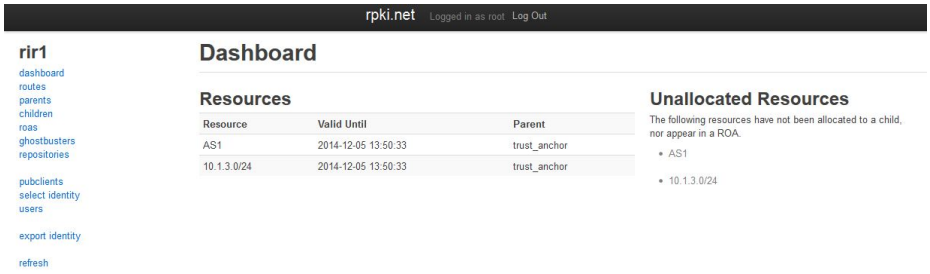

Figure 12. Les ROA définis au sein du RIR 1 (capture d'écran réalisée en date du 28/01/2014)

Comme le montre la figure 12, les ressources allouées à l'entité "rir1" sont bien "AS1" qui est en charge du sous-réseau 10.1.3.0/24. Nous pouvons également re-

marquer que dans la partie "Unallocated Ressources", l'interface web du RPKI mentionne que ces ressources n'ont pas été allouées à un fils de "rir1" et n'apparaissent pas non plus dans un ROA. La situation présentée au sein de la figure 12 est donc postérieure à la suppression du ROA qui validait "AS1" comme point d'origine d'une route vers lui-même (voir point 6.3).

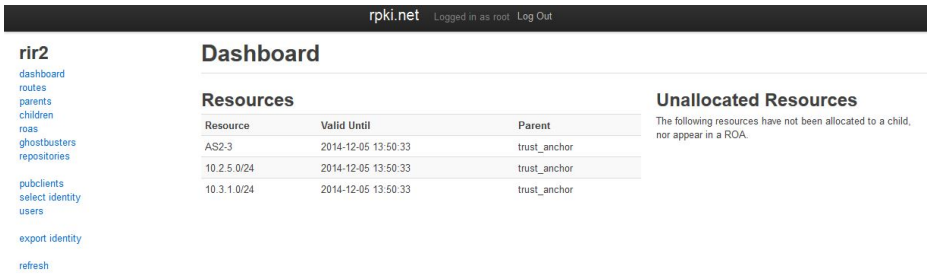

Figure 13. Les ROA définis au sein du RIR 2 (capture d'écran réalisée en date du 28/01/2014)

Comme le montre la figure 13, les ressources allouées à l'entité "rir2" sont bien "AS2" et "AS3" respectivement en charge des sous-réseaux 10.2.5.0/24 et 10.3.1.0/24. L'interface web du RPKI ne mentionne aucune ressource non allouée dans la partie "Unallocated Ressources". Les ROA créés pour "AS2" et "AS3" (qui les certifie comme point d'origine valide d'une route vers eux-mêmes) sont toujours bien en place.

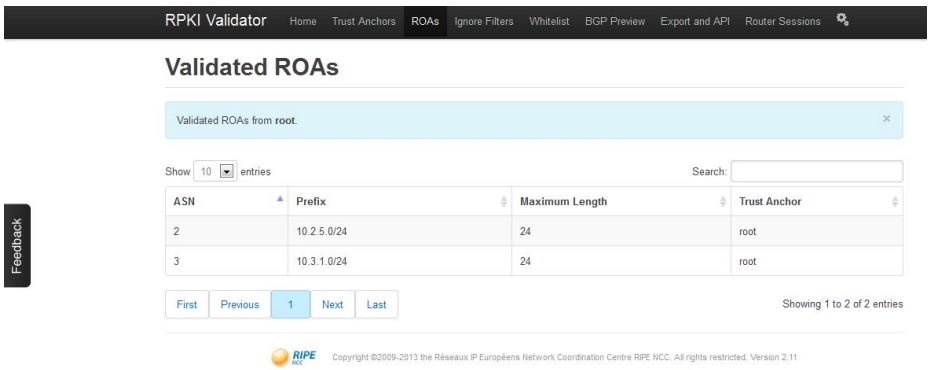

Figure 14. ROA validés par le validateur RIPE (capture d'écran réalisée en date du 28/01/2014)

La figure 14 nous montre que le validateur RIPE (qui permet de valider les objets émis par un RPKI) répercute correctement la situation puisqu'il a récupéré les deux ROA :

- le ROA de "AS2" dont le point d'origine ("ASN") est bien "AS2" et la destination ("Prefix") est 10.2.5.0/24.
- le ROA de "AS3" dont le point d'origine ("ASN") est bien "AS3" et la destination ("Prefix") est 10.3.1.0/24.

```
[2014-01-28 16:14:35,106] Client connected : /10.0.8.4:44960
[2014-01-28 16:14:35,107] /10.0.8.4:44960 -> Reset Query, hex: 00 02 00 00 00 00 00 08
[2014-01-28 16:14:35,108] /10.0.8.4:44960 <- Cache Response (session-id: -6314), hex: 00 03 E7 56 00 00 00
[2014-01-28 16:14:35,110] /10.0.8.4:44960 <- Add IPv4 Prefix (prefix: 10.3.1.0/24, maxLength: 24, Asn: AS3)
[2014-01-28 16:14:35,111] /10.0.8.4:44960 <- Add IPv4 Prefix (prefix: 10.2.5.0/24, maxLength: 24, Asn: AS2)
[2014-01-28 16:14:35,112] /10.0.8.4:44960 <- End of Data (session-id: -6314, serial: 256), hex: 00 07 E7 56
```

Figure 15. log d'envoi des messages d'update des objets validés par le validateur RIPE (capture d'écran réalisée en date du 28/01/2014)

La figure 15, elle, nous montre que le validateur RIPE envoie les messages d'update vers les différents routeurs Quagga-Srx correctement. Il envoie bien :

- un message certifiant "AS3" comme un point de départ valide vers le préfixe de destination 10.3.1.0/24. Ces informations sont envoyées lors des phases "Add IPv4 Prefix".
- un message certifiant "AS2" comme un point de départ valide vers le préfixe de destination 10.2.5.0/24. Ces informations sont envoyées lors des phases "Add IPv4 Prefix".

```
BGP table version is 0, local router ID is 10.2.5.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Validation:    v - valid, u - unknown, i - invalid, ? - undefined
SRx Status:    I - route ignored, D - SRx evaluation deactivated
SRxVal Format: validation result (origin validation, path validation)
Origin codes: i - IGP, e - EGP, ? - incomplete

   Ident    SRxVal SRxLP Status Network          Next Hop            Metric  LocPrf Weight Path
*> -------- v(v,v)              10.1.3.0/24      10.0.2.1                 0             0 1 i
*  -------- ?(?,?)          I   10.2.5.0/24      0.0.0.0                  0          32768 i
*> -------- v(v,v)              10.3.1.0/24      10.0.15.2                0             0 3 i

Total number of prefixes 3
```

Figure 16. Capture d'écran de la table de routage du pseudo-routeur BGPSec d'"AS2"

Enfin, comme le montre la figure 16, la table de routage du pseudo-routeur BGPSec d'"AS2", possède deux routes considérées comme valides :

- une route vers le préfixe 10.1.3.0/24
- une route vers le préfixe 10.3.1.0/24

L'attribut permettant de voir si une route est valide ou non au sein des tables de routage des pseudos-routeurs BGPSec est l'attribut "SrxVal". Cet attribut est formé d'un triplet c(a,b) dont "a" définit l'état de validité du ROA associé à la route et "b" l'état de validité du chemin BGPSec partant du point d'origine et arrivant à destination. La valeur "c" détermine, suivant la matrice d'état de validité des routes BGPSec (voir table 2), la validité de la route. Ces deux routes ne devraient normalement pas être "valides" puisque aucun ROA certifiant "AS2" comme point d'origine valide pour ces destinations n'existe.

Il semble donc, après examen de la situation, qu'un comportement erroné de la part de Quagga/Srx valide des routes qui normalement ne devraient pas l'être. La cause de ce comportement peut être cherchée à plusieurs endroits.

Il peut s'agir d'un mauvais fonctionnement du serveur Srx. En effet, si le serveur Srx gère la mise à jour des routes dans sa base de données de façon erronée, il envoie alors de mauvaises informations au routeur Quagga qui crée une table de routage erronée.

Il peut s'agir d'un problème de communication entre le serveur Srx et le routeur Quagga. En effet, si le serveur Srx gère correctement les mises à jour dans sa base de données mais les répercute de façon erronée au routeur Quagga, le résultat est le même : le routeur Quagga crée une table de routage erronée.

Enfin, il peut s'agir d'un problème d'interprétation des messages du serveur Srx par le routeur Quagga. Le serveur Srx gérant bien les mises à jour de routes et envoyant les bonnes données au routeur qui lui les interprète d'une façon erronée.

6.2 Absence de réaction à l'expiration des certificats

Les premiers certificats qui avaient été émis pour créer le chemin de certification du RPKI, avaient une période de validité d'un mois. Une fois ces certificats expirés, et malgré la mauvaise configuration des ROA, la mise à jour depuis le serveur RPKI aurait dû avoir pour conséquence de changer le statut de toutes les routes du testbed, faisant passer leur statut de validité de "valide" à "non valide".

Or, ce comportement ne s'est pas présenté puisque les validités de toutes les routes du testbed ont toujours gardé leur statut "valide" quelle que soit la situation.

6.3 Validité des routes malgré la suppression d'un ROA

Avant de se rendre compte de l'erreur commise dans la configuration des ROA, et afin de vérifier le bon fonctionnement du testbed, le ROA validant "AS1" a été volontairement supprimé. Cette action n'a toutefois amené aucun changement dans les valeurs de validation des routes présentes au sein du testbed.

Dans le cadre d'un testbed correctement configuré, la suppression de cet objet du RPKI aurait donc dû amener la route dont le point de départ est le sous-réseau 10.1.3.0/24 à changer de statut en passant du statut "valid" à celui de "invalid". Or, les routes partant de "AS1" ont continué à être "valid" malgré la non-existence d'un ROA certifiant "AS1" comme point de départ d'une route valide. Mais, au regard de l'erreur de configuration commise, il est difficile de déterminer quelle réaction aurait dû avoir le testbed face à ce cas de figure.

6.4 Problèmes avec le validateur RIPE

Suite à un problème technique inconnu, la première version du validateur RIPE a été altérée et il n'a donc plus été possible d'accéder à son interface et donc de vérifier s'il fonctionnait toujours correctement ou pas.

Nous avons d'abord cru, que l'altération du validateur pouvait être la cause du mauvais fonctionnement du testbed. Dès lors, suite à cet incident, une nouvelle version du validateur a été mise en place, version qui a très bien fonctionné répercutant les mises à jour de façon tout à fait correcte. Le problème ayant perduré malgré la mise en place d'une nouvelle version du validateur, nous en avons conclu que la cause du problème du mauvais fonctionnement du testbed devait se trouver en aval du validateur, c'est-à-dire au niveau de l'implémentation de Quagga/Srx.

Conclusions et perspectives

A la lumière des nouveaux éléments venus éclairer très tardivement la situation, les conclusions de ce mémoire sont assez courtes.

Comme spécifié dans la section 6, une mauvaise configuration des ROA rend quelque peu caducs les résultats des tests réalisés sur le tesbed. Néanmoins, un problème lié à Quagga/Srx et plus particulièrement à la liaison qui s'opère entre le serveur Srx et le routeur Quagga (via l'API de connexion) nous a induit en erreur quant au véritable état de validation des routes BGPSec du testbed. Il est évident que, dans un premier temps, il serait intéressant de reproduire les tests réalisés sur le testbed avec des ROA correctement configurés afin d'observer comment le testbed réagit lorsqu'il est correctement configuré.

Les perspectives futures quant à l'étude du protocole BGPSec et de Quagga sont, quant à elles, nombreuses. Au vu du problème posé par le logiciel Quagga/Srx, il serait intéressant de tester plusieurs configurations afin de pouvoir appréhender les limites de cette implémentation et voir plus précisément dans quel cadre cette implémentation peut être mise en défaut par des cas limites.

Il serait également intéressant de se pencher sur le code de Quagga/Srx afin de pouvoir pointer les parties du code qui auraient pu poser problème dans le cadre de l'implémentation mise en place pour la réalisation de ce mémoire et voir si une adaptation serait possible afin d'éviter qu'un cas de figure similaire ne se reproduise.

D'autre part, en ce qui concerne BGPSec à proprement parler, il existe un draft de l'IETF nommé "Route-leaks & MITM Attacks against BGPSec". Ce draft explique comment, au travers d'une "leak-attack", il est possible de mettre en place une attaque de type Man in the Middle (MITM) et ainsi mettre à mal la sécurisation de BGP par l'encapsulation proposée par le protocole BGPSec. Il serait intéressant d'étudier cette faiblesse dans le système de sécurisation de BGPSec et cela pourrait faire l'objet d'un travail futur.

Références

1. "http://www.iana.org". Derniere visite le 26 fevrier 2012.
2. R. Austein, G. Huston, S. Kent, and M. Lepinski. Manifests for the resource public key infrastructure (rpki). RFC 6486, Internet Engineering Task Force, february 2012.
3. Kevin Butler, Toni R. Farley, Patrick McDaniel, and Jennifer Rexford. A survey of bgp security issues and solutions. *Proceedings of the IEEE*, 98(1), Janvier 2010.
4. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, S. Boeyen, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, Internet Engineering Task Force, may 2008.
5. M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph, and R. Nicholas. Internet x.509 public key infrastructure : Certification path building. RFC 4158, Internet Engineering Task Force, september 2005.
6. R. Housley, S. Ashmore, and C. Wallace. Trust anchor format. RFC 5914, Internet Engineering Task Force, June 2010.
7. G. Huston, R. Loomans, and G. Michaelson. A profile for resource certificate repository structure. RFC 6481, Internet Engineering Task Force, february 2012.
8. G. Huston and G. Michaelson. Validation of route origination using the resource certificate public key infrastructure (pki) and route origin authorizations (roas). RFC 6483, Internet Engineering Task Force, february 2012.
9. G. Huston, G. Michaelson, and S. Kent. Certification authority (ca) key rollover in the resource public key infrastructure (rpki). RFC 6489, Internet Engineering Task Force, february 2012.
10. G. Huston, G. Michaelson, and R. Loomans. A profile for x.509 pkix resource certificates. RFC 6487, Internet Engineering Task Force, february 2012.
11. G. Huston, S. Weiler, G. Michaelson, and S. Kent. Resource public key infrastructure (rpki) trust anchor locator. RFC 6490, Internet Engineering Task Force, february 2012.
12. Geoff Huston and Randy Bush. Securing bgp with bgpsec. *The Internet Protocol Journal*, 14(2) :2–13, 2011.
13. Stephen Kent, Charlie Lynn, and Karen Seo. Secure border gateway protocol (s-bgp). *IEEE Journal on Selected Areas in Communications*, 18(4) :582–592, Avril 2000.
14. James F. Kurose and Keith W. Ross. *Computer Networking : A Top-Down Approach*. Addison-Wesley Publishing Company, USA, 5th edition, 2009.
15. M. Lepinski, A. Chi, and S. Kent. Signed object template for the resource public key infrastructure (rpki). RFC 6488, Internet Engineering Task Force, february 2012.
16. M. Lepinski and S. Kent. An infrastructure to support secure internet routing. RFC 6480, Internet Engineering Task Force, february 2012.
17. M. Lepinski, S. Kent, and D. Kong. A profile for route origin authorizations (roas). RFC 6482, Internet Engineering Task Force, february 2012.
18. M. Lepinski and S. Turner. An overview of bgpsec draft-ietf-sidr-bgpsec-overview-01.txt. draft, Internet Engineering Task Force, April 2012.

19. C. Lynn, S. Kent, and K. Seo. X.509 extensions for ip addresses and as identifiers. RFC 3779, Internet Engineering Task Force, june 2004.
20. S. Murphy. Bgp security vulnerabilities analysis. RFC 4272, Internet Engineering Task Force, January 2006.
21. M. Reynolds, S. Turner, and S. Kent. A profile for bgpsec router certificates, certificate revocation lists, and certification requests draft-ietf-sidr-bgpsec-pki-profiles-03. draft, Internet Engineering Task Force, april 2012.
22. Russ White. Securing bgp through secure origin bgp. *The Internet Protocol Journal*, 6(3) :15–22, Septembre 2003.
23. Marcelo Yannuzzi, Xavier Masip-Bruin, and Olivier Bonaventure. Open issues in interdomain routing : A survey. *IEEE Networks*, 19(6) :49–56, Novembre-Decembre 2005.