Université de Strasbourg

UFR Mathématique-Informatique

Tuteurs universitaires : Jean-Jacques Pansiot, Stéphane Cateloin et Pierre David

Sécuriser le routage sur Internet

Guillaume LUCAS

Strasbourg, le 7 mai 2013

Licence

Remerciements

Je remercie Jean-Jacques Pansiot, Stéphane Cateloin et Pierre David pour m'avoir donné l'opportunité d'étudier le présent sujet.

Je remercie vivement Stéphane Bortzmeyer pour m'avoir donné envie d'approfondir ce sujet à travers son blog ainsi que pour son suivi et ses relectures.

Sommaire

1 Introduction

Un travail sur une problématique de recherche, d'une durée approximative de quatre mois, sur un thème choisi par les étudiants, est à réaliser lors du deuxième semestre de la première année du master informatique à l'UFR Mathématique-Informatique de l'université de Strasbourg. Ce projet donne lieu à l'écriture d'un rapport et à une soutenance.

Pour ce travail, j'ai choisi de m'intéresser à la sécurisation du routage sur Internet. Il s'agit, pour moi, d'avoir une meilleure compréhension des mécanismes sous-jacents et de ses faiblesses. Il s'agit aussi de comprendre les problématiques sous-jacentes à un routage sécurisé et d'étudier, en détail, une des solutions techniques actuelles qui répond à cet impératif de sécurité : RPKI+ROA.

Mon intérêt pour cette problématique vient du fait que le routage est le pilier de tout réseau, du réseau de transport routier à Internet en passant par le réseau téléphonique. Son importance est même capitale dans les réseaux acentrés (qui n'ont pas de centre), c'est-à-dire dans les réseaux où l'information n'est pas diffusée (donc connue) depuis un unique point central du réseau, comme Internet. Compte tenu de l'importance d'Internet dans nos vies et pour nos libertés, s'intéresser à la sécurisation de son fonctionnement intrinsèque ne peut être qu'enrichissant.

Pour avoir une vision complète de RPKI+ROA, ce que ne permet pas la théorie, j'ai décidé de réaliser une maquette complète : plusieurs réseaux interconnectés, du routage dynamique entre ces réseaux et de la validation RPKI+ROA.

Ce rapport se propose de synthétiser mes recherches. Dans un premier temps, ce rapport se proposera de faire une synthèse des risques actuels. Ensuite, il fera un tour de table des différentes solutions avant de faire une présentation détaillée de l'une d'entre elles : RPKI+ROA. Enfin, la dernière partie de ce rapport se proposera de présenter ma mise en œuvre pratique de RPKI+ROA.

2 Rappels

Avant d'entrer dans le vif du sujet, je vais rappeler, brièvement, la manière dont sont faits la distribution des ressources sur Internet et le routage inter-domaine.

2.1 Adressage

Les ressources numériques uniques d'Internet, parmi lesquelles les numéros d'AS (ASN) et les préfixes IP, doivent être uniques, de manière globale, sur le réseau : dans un même temps, il ne peut pas y avoir deux machines possédant la même adresse IP publique sur Internet (sauf quand c'est désiré : cas de l'anycast, par exemple). Il faut donc distribuer ces ressources de manière cohérente au niveau mondial.

D'environ 1972 jusqu'au milieu des années 1990, cette mission était assurée par le Network Information Center. Ce NIC, qui se situait initialement au Stanford Research Institute, a été déménagé à plusieurs reprises durant les années 1990 en fonction des contrats (Network Solutions, ARIN, ...). Pour les lecteurs désireux d'en apprendre plus :

- Du temps des dinosaures, de 1972 jusqu'au début des années 1990, cette mission était assurée par le Network Information Center. Ce NIC se trouvait au Stanford Research Institute. Il assurait les "services de NIC" aussi bien auprès du Defense Data Network (ensemble mondial des réseaux militaires américains) qu'auprès du naissant Internet, descendant d'ARPANET [1]. À partir de 1988, date de création de l'IANA, cette dernière continua à lui déléguer ses responsabilités dans l'allocation des adresses et des numéros d'AS. Le NIC eut alors un rôle plus technique similaire au lien qu'il y a aujourd'hui entre l'ICANN et l'IANA.
- Dans les années 1990, les services du NIC furent assurés par différents organismes en fonction des contrats :
  - Network Solutions d'octobre 1991 à décembre 1997. De 1991 à 1993, Network Solutions était sous contrat avec Government Systems Inc. De 1993 à 1997, à la suite d'un appel d'offres, Network Solutions était sous contrat avec la National Science Foundation, dans le cadre du projet InterNIC de cette dernière. Network Solutions s'occupait de l'enregistrement des noms de domaine, de l'allocation des préfixes IP et des numéros d'AS [2].
  - De décembre 1997 à septembre 1998, l'American Registry for Internet Numbers repris les rôles attribués à Network Solutions ;
- Fin 1998, le projet InterNIC a été restructuré et la gestion des préfixes IP (v4/v6), des numéros d'AS, de la racine du DNS et des zones "reverse" a été rattachée à la nouveau-née ICANN et à l'IANA.

1. Source : `https://tools.ietf.org/html/rfc1261`.
2. Source : `http://www.nsf.gov/pubs/stis1992/nsf9224/nsf9224.txt`.

Dès le début des années 1990, face à l'augmentation du nombre de réseaux, et pour répondre à l'habituelle problématique de répartition du pouvoir, l'idée de distribuer l'allocation des ressources uniques sur Internet émerge [3].

Depuis, l'IANA (bras droit de l'ICANN pour la technique), distribue les ressources à cinq instances régionales, les Regional Internet Registry (RIR) qui sont chargés de distribuer les ressources pour une "région" du monde. Ces cinq RIR sont :

— RIPE-NCC (Réseaux IP Européens) pour l'Europe et le Moyen-Orient ;
— ARIN (American Registry for Internet Numbers) pour l'Amérique du Nord ;
— APNIC (Asia Pacific Network Information Center) pour l'Asie et le Pacifique ;
— LACNIC (Latin American and Caribbean IP address Regional Registry) pour l'Amérique latine et les îles des Caraïbes ;
— AfriNIC (African Network Information Center) pour l'Afrique.

Les RIR distribuent à leur tour les ressources qu'ils obtiennent selon leur propre politique d'allocation. Les ressources indépendantes (numéro AS, préfixes IP dits Provider-Independent, PI), sont distribuées sur justificatif d'usage prévu, directement aux opérateurs réseaux. Les autres ressources (préfixes IP dits Provider-Aggregatable, PA) sont distribuées aux membres des RIR, appelés Local Internet Registry (LIR) [4].

Les LIR utilisent les ressources qu'ils obtiennent pour leur usage propre mais ils peuvent aussi distribuer une partie de leur allocation, à un niveau local, à d'autres opérateurs, sur justificatif d'usage prévu (information à transmettre au RIR).

Il y a néanmoins quelques exceptions à cette distribution :

— Certaines ressources sont réservées à des usages précis (exemples : documentation, adresses privées, 6to4, ...) et ne sont donc pas distribuées [5].
— Les préfixes obtenus avant l'apparition des RIR, appelés legacy (l'héritage) ou swamp (le marais), sont en dehors de ce système de distribution et des "règlementations" associées. Au début des années 2000, les RIR se sont réparti ces préfixes en fonction de la localisation géographique de leur titulaire : c'est le programme ERX - Early Registration Transfer [6]. Les RIR tentent de convaincre les titulaires de ressources legacy de devenir membres et de participer ainsi aux frais de gestion [7].
— Dans certaines régions du monde (surtout en Asie et en Amérique Latine), on trouve également des National Internet Registry (NIR), c'est-à-dire des entitées qui font partie de la hiérarchie

3. Source : `https://tools.ietf.org/html/rfc1174`.
4. Source : `https://www.ripe.net/ripe/docs/ripe-553`.
5. Voir : `https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml`, `https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xml` ou bien encore `https://www.iana.org/assignments/as-numbers/as-numbers.xml`.
6. Voir, par exemple : `https://www.ripe.net/lir-services/resource-management/erx`.
7. Voir, par exemple `https://services.renater.fr/connectivite/erx`.

d'allocation des ressources uniques au même niveau qu'un LIR (donc sous "l'autorité" d'un RIR) mais qui agissent à un niveau national ou pour le compte d'une entité économique bien définie. Exemple : China Internet Network Information Center ou Network Information Center Mexico.

2.2 Routage inter-domaines

Le protocole de routage utilisé entre les AS pour s'échanger des routes et construire dynamiquement les tables de routage est BGP. Sa version 4 (celle qui doit être utilisée) a été normalisée, pour la première fois, dans le RFC 1654 en 1994 et mis à jour depuis (RFC 1771 en 1995 et RFC 4271 en 2006). BGP est catégorisé comme un protocole à vecteur de chemins.

L'échange des routes se fait au-dessus d'une session TCP, sur le port 179. Les échanges entre deux pairs BGP (deux routeurs interconnectés directement) se font à l'aide de différents messages tels que « OPEN » pour ouvrir la connexion, négocier les options et s'identifier (simple échange des numéros d'AS), « NOTIFICATION » pour clôturer une session si une erreur a été détectée, « UPDATE » pour annoncer l'ajout ou le retrait d'une ou plusieurs routes.

Vu que l'on est dans de l'échange inter-AS, entre entités distinctes voire plus, il n'y a aucune raison que les AS tombent d'accord pour utiliser une même route pour un même préfixe. Plus globalement, le choix d'une route en fonction de critères subjectifs (on ne veut pas passer par tel AS car on a des doutes le concernant, on veut passer en priorité par tel AS car on a un contrat de transit moins cher, ...) est au cœur du métier d'opérateur. Contrairement aux protocoles de routage internes, BGP prend en compte cette spécificité. Par exemple, le choix de pouvoir envoyer des informations à tel AS mais pas à un autre est une fonctionnalité de base de BGP et celle qui est la plus utilisée alors qu'elle n'existe pas (et n'a pas de sens) dans les protocoles de routage dynamique intra-AS.

Une autre caractéristique de BGP est qu'il est un protocole qui se base sur la confiance en la bienveillance des AS : n'importe quel AS peut annoncer ou relayer n'importe quelle information, typiquement une route, sans que l'information ne soit authentique, c'est-à-dire sans que l'annonceur ou le relayeur dispose bel et bien d'une route pour le préfixe qu'il annonce. Il s'agit du problème racine de BGP, celui qui conduit à chercher une solution de sécurisation depuis de nombreuses années.

3 De l'intérêt d'un routage sécurisé

Le risque d'une attaque sur le routage Internet est connu depuis le début des années 1990 (la DARPA a missionné la société BBN pour réfléchir à cette problématique et tenter d'y apporter une solution). Nous allons d'abord nous intéresser aux attaques qu'il est possible de mener sur BGP et à leurs conséquences puis nous verrons quelques attaques concrètes.

3.1 Quels dommages ?

Avant de présenter les différentes attaques qui visent le protocole BGP, intéressons-nous au résultat de ces attaques : quels dommages peuvent-elles engendrer ?

Le RFC 4272 « BGP Security Vulnerabilities Analysis » [Murphy(2006)] en fait un résumé :

— Un attaquant peut vouloir empêcher ou ralentir l'accès à un préfixe précis. On peut dès lors imaginer toutes sortes de motivations : simple attaque "pour s'amuser", vengeance, attaque économique contre un concurrent dont les affaires se font majoritairement sur le web, ...

— Il peut aussi vouloir provoquer un ralentissement d'une partie du réseau en créant de la congestion, une fluctuation incessante des routes (d'où un problème de convergence et d'épuisement des ressources des routeurs pour traiter les messages BGP), ...

— Un attaquant peut vouloir rediriger des flux de données vers son réseau afin de les espionner voire de les modifier avant de les re-émettre vers leur destination d'origine. Les motivations principales sont l'intelligence économique et la collecte d'informations.

— Il peut aussi mener des attaques actives de plus grandes envergures sur les protocoles de transport/applicatif puisqu'il est en mesure de se faire acheminer des données qui ne lui sont originellement pas destinées et d'émettre depuis des adresses qui ne lui sont pas attribuées.

Le RFC propose une classification plus fine (deux attaques ayant pourtant le même but seront différenciées) mais pour résumer, les attaques BGP ont, principalement, trois objectifs (une attaque peut cibler un ou plusieurs de ces objectifs) :

— Confidentialité : intercepter des flux de données, comprendre les politiques de routage d'un AS, obtenir des informations pour des attaques sur les couches hautes, ...

— Intégrité : modifier des données dans les flux interceptés, ...

— Performance : empêcher ou ralentir l'accès à des préfixes, surcharger des liens/des routeurs.

3.2 Quelles attaques ?

Les travaux d'Antoine Gallais [Gallais(2012)], de Joseph E. Fares [Fares(2004)], de Christian Horn [Horn(2009)] ainsi que l'étude « A Study of Prefix Hijacking and Interception in the Internet » [Hitesh Ballani(2007)] m'ont aidé pour la rédaction de cette sous-partie en plus des autres documents cités.

Il y a deux grandes catégories d'attaque contre BGP :

D'une part, on peut s'attaquer à la communication entre deux pairs BGP. La communication entre deux pairs BGP se faisant sur TCP, BGP est sensible aux attaques qui concernent TCP (vol/découverte des numéros de séquence, reset de la connexion, IP spoofing, ...). Cette catégorie d'attaque est hors de mon sujet car plusieurs solutions (BCP, sécurité du canal, ...) ont déjà été apportées et sont déployées.

D'autre part, on peut injecter de fausses informations qui, comme les informations légitimes, se propageront de pair BGP en pair BGP. Nous allons détailler les attaques possibles.

3.2.1 Détournement d'un préfixe (prefix hijack)

Un attaquant annonce être à l'origine d'un préfixe déjà annoncé. L'annonce se propagera et l'attaquant récupèrera une partie du trafic de sa victime. Une partie seulement car le préfixe étant annoncé par plusieurs AS, le trafic sera réparti entre les différents AS qui annoncent le préfixe en fonction de la longueur du chemin et des politiques de routage. C'est d'ailleurs ce principe qui est utilisé dans l'anycast. Dans le même ordre d'idée, il est tout à fait possible d'annoncer des préfixes qui n'ont pas encore été alloués. Par exemple : le préfixe 42/8 n'a été distribué, par l'IANA, qu'en octobre 2010. Pourtant, ce préfixe (ou un sous-préfixe) a été annoncé par plusieurs opérateurs dont Level 3 en 2004, Telefonica Espagne en 2007, le département de la défense américaine en 2008, ... [8] Tous ces cas ont été des erreurs rapidement corrigées. Pourtant, un attaquant peut annoncer un préfixe non alloué et s'en servir de manière malveillante (pour du spam, par exemple). Cela permet aussi de nuire au futur titulaire du préfixe : dans un cas de spam, par exemple, le titulaire devra faire supprimer son préfixe des listes noires [9]. Il faut nuancer ce dernier point en disant qu'il ne reste plus de blocs v4 non alloués et que les annonces portant sur des blocs v6 non alloués sont souvent filtrées par les opérateurs.

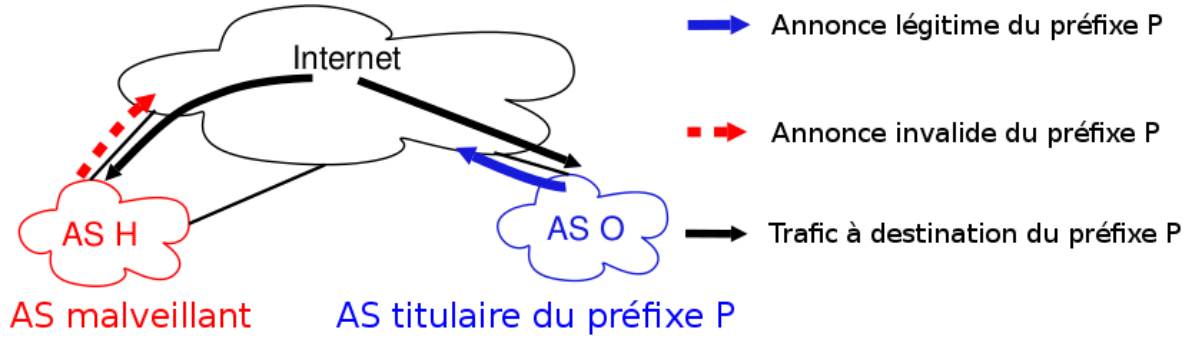

FIGURE 1 – Détournement d'un préfixe. D'après [Hitesh Ballani(2007)].

8. Ces faits peuvent être vérifiés avec l'outil REX du RIPE : `http://albatross.ripe.net/cgi-bin/rex.pl?res=042%2F8&type=all&stime=2000-08-20&etime=2010-09-30&cf=1&af=1`.

9. Une utilisation frauduleuse du préfixe non alloué 2.0.0.0/12 pour, entre autres, ce qui semble être du phishing a été détecté et documenté : [François CONTAT(2012)].

3.2.2 Détournement d'un sous-préfixe

Même principe que précédemment, mais cette fois-ci, l'attaquant annonce un sous-préfixe. Le préfixe étant plus spécifique que celui de l'annonce légitime, le trafic à destination de ce sous-préfixe sera transporté intégralement jusqu'à l'attaquant. Selon l'organisation du réseau que l'on veut usurper, une annonce en sous-préfixe est suffisante. Imaginons que toutes les machines hébergeant le service A d'une entreprise sont toutes dans le réseau 198.18.1.0/24. L'entreprise annonce le préfixe qui lui a été alloué à la base, 198.18.0.0/16 mais on voit bien qu'annoncer le sous-préfixe /24 permet d'atteindre les mêmes objectifs si l'attaquant vise uniquement le service A. Dans le même ordre d'idée, si un sous-préfixe n'est pas utilisé par le titulaire d'un préfixe, un attaquant peut l'annoncer et s'en servir de manière malveillante (pour du spam, par exemple) sans compromettre les activités du titulaire du préfixe et donc être furtif plus longtemps.

3.2.3 Modifier le chemin d'AS

Il est possible d'ajouter ou de retirer des AS du chemin d'AS. Si un attaquant veut seulement prendre connaissance d'un flux (et donc le transmettre à la destination), il peut s'insérer dans le chemin d'AS, en prenant garde à être connecté en direct avec les AS qui l'entoure dans le chemin d'AS. Si un attaquant veut détourner/rendre inaccessible un (sous-)préfixe sans modifier l'origine, il peut modifier le chemin d'AS pour s'annoncer comme étant l'AS frontal, la seule porte d'entrée/sortie de l'AS d'origine. L'ajout d'AS aux chemins permet aussi de déclencher des détections de boucles et de ralentir (voire d'empêcher) le transport d'un flux d'information. Le problème de la modification du chemin d'AS, surtout de l'ajout, est qu'une route avec un AS supplémentaire dans sa chaîne ne sera pas sélectionnée par la majorité des routeurs (l'un des critères de choix principal est la longueur du chemin d'AS). Au préalable, l'attaquant peut annoncer le retrait de la route d'origine dans un message BGP UPDATE mais il sera vite détecté. La difficulté est donc de construire un chemin qui soit considéré comme étant intéressant par le nombre de routeurs nécessaires à l'atteinte de l'objectif de l'attaque tout en étant valide (la validité dépend de l'objectif de l'attaque) et acceptable.

3.2.4 Nuire aux performances

Pour atteindre cet objectif, il est possible d'annoncer des préfixes qui ne doivent pas être annoncés sur Internet : RFC 1918, documentation, préfixes non distribués par l'IANA. Cependant, les bogons (préfixes non distribués par l'IANA) sont souvent filtrés. Il est aussi possible d'annoncer tous les préfixes désagrégés, c'est-à-dire, en IPv4, annoncer tous les /24 possibles (les annonces plus précises sont majoritairement filtrées). Cela s'est produit en 1997 et a donné lieu à une fonctionnalité de contrôle du nombre maximum de préfixes qu'un pair peut envoyer dans un intervalle de temps. Il est aussi possible de provoquer des congestions en faisant fluctuer les routes : ajout et retrait, très

rapidement, afin de forcer les routeurs à actualiser leur table de routage. Un algorithme permettant de limiter la fluctuation a été élaboré et normalisé (RFC 2439). Cet algorithme prévoit de supprimer un préfixe de la table de routage en attendant que la fluctuation s'arrête. On s'est donc rendu compte que cet algorithme permet de faire un déni de service : il suffit de faire fluctuer la route d'un tiers pour qu'elle soit rejetée. Si le RFC n'est pas obsolète, le groupe de travail sur le routage du RIPE ne recommande pas son utilisation [10]. Enfin, il est possible de construire un chemin d'AS tel qu'il fasse passer une quantité de données par des routeurs qui ne peuvent pas supporter une telle charge de manière à les surcharger et provoquer un déni de service. On retombe ici dans la problématique de construire un chemin d'AS qui soit accepté tout en permettant de mener à bien l'attaque. Pour nuire aux performances, il est aussi possible de forger des messages BGP inattendus. Cela provoquera un comportement inattendu (arrêt de la session BGP, surcharge, ...) d'un certain nombre de routeurs en fonction de leur programmation. Plusieurs cas se sont déjà produits, nous y reviendrons. On peut nuancer ce point en disant qu'on ne trouve pas de tels bugs tous les jours. Néanmoins, comme certaines sociétés travaillent à la conception de matériel offensif de type 0-day [11], je pense que ce genre d'attaque n'est pas improbable.

3.2.5 Anton Kapela & Alex Pilosov

Lors de la DEFCON 16, à l'été 2008, les chercheurs Anton Kapela et Alex Pilosov ont présenté une nouvelle méthodologie pour une attaque BGP visant à intercepter le trafic de manière plus silencieuse. [DEF(2008)]

L'attaque se déroule en plusieurs étapes :

— Premièrement, l'attaquant doit observer les chemins disponibles vers l'AS qu'il cible. Il devra alors choisir celui du retour. Sur le schéma ci-dessous, le chemin de retour choisi sera AS10 - AS20 - AS200.

— Deuxièmement, l'attaquant doit fabriquer la fausse annonce BGP. D'une part, annoncer un sous-préfixe de la cible et d'autre part, utiliser la technique de l'AS-path prepending pour ajouter les AS du chemin de retour à l'annonce. Cette technique, combinée à celle du détournement de sous-préfixe fera en sorte que l'annonce soit acceptée par une majorité de routeurs sauf ceux choisis pour le chemin de retour (pour cause de détection d'une boucle). Dans notre cas, l'annonce portera sur le préfixe 10.10.220.0/24 (le premier sous-préfixe de 10.10.220.0/22) et il faudra ajouter « 10 20 200 » au chemin d'AS.

— Troisièmement, l'attaquant ajoutera une route statique sur son routeur indiquant que le trafic capturé pour le préfixe usurpé doit être transmis via le premier AS du chemin de retour. Dans

10. Voir la recommandation du groupe de travail du RIPE : `https://www.ripe.net/ripe/docs/ripe-378`.

11. On pourra citer, par exemple, la société française Vupen Security qui a une éthique mouvante maintes fois décriée. Par exemple : `http://reflets.info/aurora-hackers-chinois/`.

notre cas, l'AS 10 est le premier AS du chemin de retour et 10.10.220.0/22 le préfixe recouvrant. Il faudra donc ajouter une route statique "10.10.220.0/22 via 1.2.3.4" (ou 1.2.3.4 est l'un des routeurs de bordure de l'AS 10 avec lequel l'AS attaquant est connecté).

— Enfin, l'attaquant falsifiera le TTL des paquets qu'il retransmet afin de masquer son attaque (traceroute repose sur le TTL, le diagnostic premier et "humain" d'une usurpation se fait, majoritairement avec traceroute). Des morceaux de codes sont disponibles.

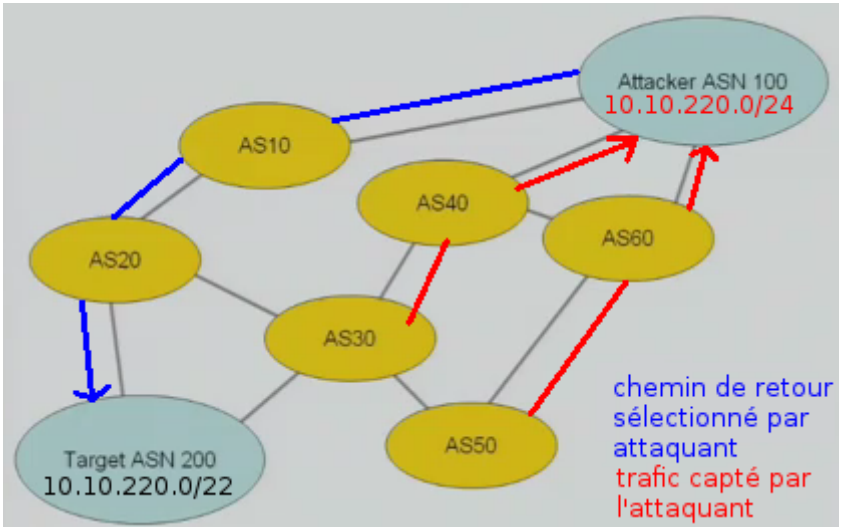

FIGURE 2 – Kapela et Pilosov. D'après les slides de Kapela et Pilosov à la DEFCON 16.

Comme nous pouvons le constater, cette attaque n'est pas révolutionnaire : elle n'est que la combinaison de techniques (détournement de sous-préfixe, as-path prepend, ..) largement connues. L'innovation de Kapela et Pilosov est de rendre accessible un détournement de préfixe tout comme Kaminsky a mis l'empoisonnement de cache DNS à la portée de tous.

Bien évidemment, l'attaque sera détectée par les systèmes d'alerte BGP (nous y reviendrons) mais passera inaperçue pour le réseau cible et pour ses utilisateurs puisque le trafic est acheminé. Donc si le réseau cible n'est pas abonné à un système d'alerte, sa découverte de l'attaque le visant prendra du temps.

3.3 Des attaques concrètes

Aucune attaque n'a jamais été publiquement détectée et/ou annoncée mais de nombreux incidents, dus principalement à des erreurs humaines se sont produits et ont été rendus publics par leur ampleur. Parmi eux (liste **non exhaustive**) :

— 1997 : un routeur de l'AS 7007 (MAI Network Services) désagrège tous les préfixes qu'il reçoit en préfixes de longueur /24. Les préfixes étant plus spécifiques, les annonces invalides se propagent, ce qui cause un trou noir pour les AS ayant appris les annonces incorrectes.

— 1998 : l'AS 8584 annonce lui aussi être à l'origine de tous les préfixes qui composent Internet.

— 2001 : l'AS 15412 annonce des routes qu'il n'a pas.

— 2004 : en mai, l'hébergeur malaisien DataOne a usurpé deux préfixes de Yahoo. En décembre, le fournisseur d'accès turc TTNET a réannoncé l'ensemble des routes de l'Internet.
— 2005 : Cogent annonce un préfixe de Google.
— 2006 : en janvier, la société américaine Con Edison annonce des préfixes qu'elle ne détient pas.
— 2008 :
  — en février, Pakistan Telecom, répondant à une volonté du gouvernement de bloquer l'accès à Youtube, annonce des préfixes plus spécifiques que les préfixes originaux de Youtube. PCCW, son transitaire, n'a pas filtré les annonces qui se sont donc propagées et ont ainsi créé un trou noir pour certaines personnes voulant accéder à Youtube.
  — En mars, Abovenet annonce les préfixes du FAI Africa Online Kenya.
  — En mai, Telianet annonce la moitié d'Internet en un seul bloc (0.0.0.0/1).
  — En novembre, le FAI brésilien CTBC a réannoncé l'intégralité des préfixes mais sans conséquence extérieures (simple isolation de CTBC par ses pairs) car les pairs BGP ont filtré les annonces.
— 2010 : en avril, l'opérateur China Telecom annonce environ 37 000 préfixes (soit environ 11 % des préfixes annoncés à cette période) qu'il ne détient pas.
— 2012 : l'opérateur australien Telstra a réannoncé l'intégralité des préfixes à la suite d'une erreur d'un de ses clients mais sans conséquence extérieure (isolation de Telstra par ses pairs) car les pairs BGP ont filtré les annonces.
— 2013 : l'opérateur Kazakhtelecom annonce 89.0.0.0/8. Il s'agit d'une annonce trop large Voir `http://www.bortzmeyer.org/annonces-bgp-larges.html`) qui englobe des préfixes alloués aux opérateurs Free, Alsace Réseau Neutre, ...

Tous ces exemples à l'exception du dernier, que j'ai vécu, sont issus du cours d'Antoine Gallais [Gallais(2012)] ainsi que de la conférence « Influence des bonnes pratiques sur les incidents BGP » présentée au SSTIC 2012 par l'ANSSI/Orange [François CONTAT(2012)].

Il est intéressant de s'interroger également sur les incidents/attaques (ciblés ou non) qui n'ont pas été d'une grande ampleur et n'ont donc pas été rendus publics ou seulement dans les milieux spécialisés (exemples : 42/8 évoqué précédemment, OSIRIS [12], « Influence des bonnes pratiques sur les incidents BGP » du SSTIC 2012, ...) : quels en sont la fréquence et l'impact ?

Néanmoins, il faut relativiser : des facteurs limitants font qu'on n'a pas une attaque contre le routage inter-domaines par jour. Parmi ces facteurs :

— Les attaquants ont besoin d'Internet donc le paralyser totalement est une mauvaise idée.
— Il y a une différence entre « vulnérabilité découverte, documentée et présentée dans les conférences orientées sécurité » et « vulnérabilité exploitée » : les attaquants sont rationnels et

12. OSIRIS, le réseau métropolitain de Strasbourg pour l'enseignement supérieur et la recherche, a subi un détournement de préfixe (accidentel, semble-t-il) dans les années 1990.

choissisent toujours les attaques qui ont le meilleur rapport efficacité/coût et jamais les attaques hyper complexes (sauf attaques ciblées). Il n'y a qu'à voir la bétise grasse des mails de phishing ou le non respect du protocle SMTP par les spammers dans le but d'aller vite (ehlo). Un coût élevé pour une fenêtre d'intervention étroite, ce qui signifie un profit limité, c'est, selon moi, ce qui explique l'absence d'attaques sérieuses et répétées contre le routage Internet. De plus, les vulnérabilités de BGP sont connues : cela pourrait permettre des attaques mais cela permet aussi une plus grande résistance : les attaques étant connues, la communauté Internet [13] surveille ces portes d'entrée.

— Coût (financier/administratif/technique) de mise en place. Soit il faut pirater des routeurs en bordure d'AS, soit il faut mettre en place le sien ce qui, au minimum, requière d'avoir des contrats avec des pairs BGP.

— Dès que l'attaque/incident sera détectée, la réactivité humaine fera que des contre-mesures seront mises en place. La fenêtre d'attaque est donc réduite, même avec les optimisations de Kapela&Pilosov.

— Le fait que les annonces soient publiques, donc visibles de tous, empêche les acteurs sérieux (ceux qui ont un business, une visibilité, une attache géographique) de commettre ou de laisser faire une attaque.

— Selon les objectifs (vol de données, déni de service, ...), des méthodes plus efficaces et moins "couteuses" existent pour y parvenir :

  — Si un état veut faire de l'intelligence économique visant d'autres états, il lui suffit de demander à ses diplomates d'espionner les diplomates des autres états [14]. Si un état veut mettre sur écoute tout ou partie de sa population ou faire de l'intelligence économique, il existe une solution plus intéressante et efficace sur le long terme : le Deep Packet Inspection [15].

  — En ce qui concerne le déni de service (distribué ou non), la baisse de performances ou la saturation de liens, sa mise en œuvre est tellement aisée sur les couches hautes [16] qu'il me semble inutile d'attaquer le routage, plus complexe à détraquer pour une détection tout aussi immédiate.

Pour résumer : à l'heure actuelle, aucune attaque n'a été rendue publique. Nous savons, en revanche, que de nombreux incidents se sont produits. C'est pour cette raison qu'il me semble important de consolider le routage Internet ou, en tout cas, d'y réfléchir : les déployements sur l'Internet sont longs.

13. Il ne reste plus qu'à définir ce terme. :)

14. Hillary Clinton, en sa qualité de secrétaire d'état, avait demandé aux diplomates américains d'espionner les autres diplomates et bien plus. Voir : `http://www.lemonde.fr/documents-wikileaks/article/2010/12/08/wikileaks-colere-feutree-a-l-onu-apres-les-accusations-d-espionnage_1450880_1446239.html`.

15. Le DPI a très bien fonctionné en Libye pour capturer et torturer des opposants au régime de Kadhafi, fonctionne très bien en Syrie et fonctionnera très bien au Maroc et peut-être un jour en France. Voir : `http://reflets.info/category/sagas/deep-packet-inspection/`.

16. On se rappellera, entre autres, de l'opération Payback, lancée en décembre 2010 par les Anonymous et les sympathisants, visant à "venger" les attaques contre Wikileaks. Cela avait abouti à un déni de service de grandes institutions comme MasterCard, Visa et la banque PostFinance.

4 Quelles solutions ?

4.1 Historiques

Le travail de Laurent Charbonnier [CHARBONNIER(2007)] m'a aidé pour la rédaction de cette sous-partie en plus des autres documents cités.

4.1.1 Systèmes d'alarme

Une des solutions aux attaques BGP est la surveillance via des systèmes d'alarme. Il s'agit de surveiller la plus grande part possible d'annonces BGP depuis un ou plusieurs points sur Internet (principe de dispersion des sondes) et de reporter les annonces suspectes (un AS supplémentaire annonce un préfixe déjà annoncé, un AS s'annonce comme transitaire supplémentaire d'un AS pour un préfixe donné, ...).

Il en existe un grand nombre : du service Information Services Alarm du RIPE au service Routing Alarms de Renesys en passant par le service Route Monitoring de BGPMon [17].

Des papiers universitaires détaillent des compléments à cette méthode qui permettent d'éviter les faux-positifs. Par exemple : en cas d'annonce BGP différente de celles vues précédemment, on peut scanner le réseau d'origine afin de réaliser une prise d'empreintes. Si elles sont différentes de celles archivées : il y a réellement une attaque. Dans le cas contraire, l'annonce est valide malgré tout. Je reste néanmoins sceptique : il y a très peu de cas où cela peut être efficace. En cas d'usurpation de préfixe, ce n'est que perte de temps, la première détection sera bonne. Dans le cas d'un chemin d'AS modifié, l'origine peut être bonne mais comment être sûr que le transitaire inséré n'est pas malveillant ?

D'autres papiers universitaires détaillent une méthode de détection basée sur le nombre de sauts de réseau pour atteindre le réseau final depuis une sonde. Ce nombre de sauts est, généralement, plutôt constant. Une forte fluctuation de ce nombre de sauts indique donc un problème.

Les avantages des systèmes d'alarme sont qu'ils sont faciles à utiliser et qu'ils ont fait leurs preuves à plusieurs reprises.

L'inconvénient principal est que les systèmes d'alarme sont passifs car basés sur la détection d'un problème. Quand la détection est faite, le mal est déjà fait. Il faut prévenir les responsables du réseau concerné et prendre des contre-mesures en faisant attention, sous l'effet du stress, de ne pas déployer une solution pire que le problème.

Le deuxième inconvénient est qu'il est impossible de couvrir tous les Internets avec un réseau de sondes. Ces derniers ont donc une visibilité limitée. Il existe donc de nombreuses zones d'ombres dans lesquelles des attaques peuvent survenir en toute furtivité d'un point de vue des systèmes d'alarme.

17. Pour l'IS Alarm du RIPE, voir : `http://meetings.ripe.net/ripe-57/presentations/Schwarzinger-MyASN.dOuq.pdf`. Pour Renesys, voir : `http://www.renesys.com/products_services/routing_alarms/index.shtml`. Pour BGPMon, voir : `http://www.bgpmon.net/services/route-monitoring/`.

4.1.2 BCP

Les Best Current Practice (BCP) sont un ensemble de bonnes pratiques qui deviennent, en standard, la bonne manière de mettre en place un système. Notons que l'IETF n'a publié qu'un seul BCP concernant les communautés BGP. Mais il existe un ensemble de bonnes pratiques concernant BGP qui sont majoritairement admises :

- Authentifier les pairs lors d'un échange (MD5, TCP-AO, IPSec), sur une interface réseau secondaire, en fixant les IP et les adresses MAC. Dans la même veine, penser à la "sécurité TTL" : le TTL est fixé à 255 en sortie, et, sachant que, dans le cas simple, deux pairs BGP sont directement connectés (même L2), détruire les paquets qui ont un TTL différent de 255 en réception.
- Filter les martians (blocs réservés (RFC1918, documentation, tests, ...), numéros d'AS réservés), les bogons (préfixes non alloués par l'IANA), les annonces des clients (ils ne doivent annoncer que leurs préfixes) voire filtrer en suivant les déclarations dans les IRR (Internet Routing Registry).
- Rejetter les annonces trop larges ($< /8$ en IPv4) ou trop spécifiques ($> /24$ en IPv4, $> /48$ en IPv6).
- Définir des seuils pour éviter les surcharges (ex : maximum-prefix).

On notera que, dans la réalité, ces BCP sont suivies de manière fluctuante : les incidents concrets nous ont montré que les annonces des clients ne sont pas toujours filtrées (PCCW/Youtube, CTBC, Telstra, ...). Les bogons sont trop ou pas assez filtrés selon les cas, il n'y a qu'à voir sur les listes de discussion orientées réseau. Certains opérateurs sont néanmoins (re)connus pour leur sérieux à ce sujet (exemple : Level 3 filtre selon les IRR).

C'est pour cela que, à mon avis, les BCP ne représentent pas une approche suffisante pour sécuriser le routage Internet.

4.1.3 S-BGP

Secure-BGP (S-BGP) [Charles Lynn(2003)] [Kent(?)] est une solution développée par BBN Technologies, dès le milieu des années 1990, sous contrat avec la DARPA et dont le concept est de vérifier l'origine et le chemin d'AS d'une annonce en utilisant massivement la cryptographie. S-BGP est la première des solutions proposées à traiter intégralement tous les problèmes de sécurité du routage BGP (authentification de l'origine, des routeurs intermédiaires et des pairs entre eux).

Pour l'authentification des pairs entre eux, S-BGP impose l'utilisation d'IPsec, solution de sécurité du canal normalisée mais très peu utilisée.

Pour les problématiques liées à l'authentification du message, S-BGP propose une infrastructure à clés publiques : les instances qui allouent les numéros d'AS (IANA/RIR) et les préfixes

(IANA/RIR/LIR) distribueront les certificats qui permettront d'associer un opérateur à(aux) ASN et au(x) préfixe(s) qu'il détient.

Pour authentifier l'origine de l'annonce, c'est-à-dire pour savoir si l'AS qui se prétend être à l'origine d'un préfixe a bien le droit d'annoncer ce préfixe, S-BGP propose une Address Attestation, objet signé par le titulaire d'un préfixe qui permet d'indiquer quels AS sont autorisés à s'annoncer comme étant l'origine du préfixe.

Pour authentifier chacun des routeurs intermédiaires dans le chemin d'AS, S-BGP propose une Route Attestation, objet signé par un routeur du chemin contenant les numéros d'AS du chemin actuel ainsi que ceux des pairs à qui l'annonce sera envoyée. Par exemple : pour un chemin d'AS : 64503 64502 64501, 64502 créera et signera une RA(64503, 64502, 64501).

Pour valider une route, un routeur effectue donc plusieurs opérations. D'une part, il vérifie l'origine de l'annonce avec l'Address Attestation et sa signature. Ensuite, il vérifie la signature de chaque Route Attestation (la signature a bien été généré par l'AS que le routeur prétend représenter). Puis, il vérifie qu'il possède autant de Route Attestation que de routeurs traversés. Enfin, il vérifie qu'il y a une correspondance entre chaque AS dans le chemin et ce qui est annoncé dans la Route Attestation correspondante.

Cela fait beaucoup d'objets à transmettre. S-BGP propose de stocker les clés publiques, les certificats, les Address Attestation et les listes de révocation dans des dépôts. Ces dépôts pourront être tenus par les RIR, les grands opérateurs voire par de petits opérateurs et se synchroniseraient entre eux. Pour les Route Attestation, il n'y a pas le choix, elles doivent être transmises via BGP lui-même à travers un nouvel attribut.

Avantage :

— S-BGP est une solution complète.

— S-BGP ne nécessite pas une modification du protocole BGP.

Inconvénients :

— S-BGP est complexe à mettre en place en une seule étape.

— S-BGP induit une surcharge de consommation mémoire ainsi qu'une surcharge du CPU dans les routeurs BGP. Ceci est inévitable : la cryptographie nécessite des calculs supplémentaires. Stephen T. Kent, un des concepteurs indiquait même que l'infrastructure actuelle ne permettait pas de déployer S-BGP à cause de ces deux facteurs.

— S-BGP reste encore vulnérable à une attaque par rejeu (annoncer une route qui a été retirée). Il faut nuancer ce point en affirmant que cette attaque n'est pas triviale et qu'elle est limitée dans le temps.

4.1.4 soBGP

Secure Origin BGP (soBGP) [White(2005)] [White(2003)] [White(?)] est une solution proposée en 2003 par Cisco dont le concept est de vérifier, de manière cryptographique, pour chaque pair BGP, s'il est apte à annoncer une route, c'est-à-dire, s'il a bien au moins un chemin valide jusqu'à la destination qu'il annonce. Cela inclut le fait de savoir si un AS peut être à l'origine d'un préfixe qu'il annonce. SoBGP a fait l'objet de plusieurs brouillons à l'IETF.

SoBGP utilise une panoplie de certificats x509 :

— EntityCert : fait un lien entre un AS et les parties publiques de ses clés. Comment être sûr d'une bonne association ? SoBGP n'utilise pas une infrastructure à clés publiques mais un réseau de confiance. Un faible nombre de clés distribuées en dehors de BGP et détenues par des autorités de confiance (grands opérateurs ? les CA x509 classiques ?) pourraient signer des EntityCert. Les clés privées associées à un EntityCert signé pouvant, à leur tour, signer d'autres EntityCert.

— AuthCert : fait le lien entre un AS et les préfixes qu'il détient. Signé par la partie privée d'une des clés publiques contenues dans l'EntityCert.

— PolicyCert : possibilité, pour un AS, de définir des politiques pour ses préfixes. Exemples : « la longueur maximale qui pourra être annoncée est /24 », « je ne veux pas que tel AS soit dans le chemin d'AS ». Signé par la partie privée d'une des clés publiques contenues dans l'EntityCert.

— ASPolicyCert : fait le lien entre un AS et ses pairs : peer, transit, ... Signé par la partie privée d'une des clés publiques contenues dans l'EntityCert.

Tous ces certificats sont diffusés grâce à un nouveau message BGP : SECURITY. Il y a donc modification du protocole BGP.

Pour détecter une annonce invalide, un routeur se basera sur l'AuthCert. Pour détecter un chemin invalide, il se basera sur l'ASPolicyCert. Si le chemin indique 64501 64503 64502, il faut que l'ASPolicyCert de l'AS 64501 indique qu'il a un lien direct avec l'AS 64503, et réciproquement et que l'ASPolicyCert de l'AS 64503 indique qu'il a un lien direct avec l'AS 64502 et réciproquement. Si, par erreur, l'AS 64501 indique un chemin : 64501 64502 (et met à jour son ASPolicyCert), le chemin sera refusé car l'AS 64502 ne déclare pas avoir un lien direct avec l'AS 64501.

Avantage de SoBGP : il ne repose pas sur une PKI complexe à mettre en place.

Inconvénients :

— SoBGP est typique des solutions prévues pour tous les usages : le brouillon de standard indique en effet qu'une infrastructure à clé publique pourra être utilisée plus tard et que l'échange des certificats et la cryptographie attenante pourront être pris en charge par des machines externes aux routeurs ... plus tard.

— Modification du protocole BGP.

— À mon avis : solution complexe vue le nombre de certificats x509 en présence.

— Problèmes habituels des réseaux de confiance : expiration des certificats ? Roulement des clés ? Révocation ?

4.1.5 psBGP

Pretty Secure BGP (psBGP) [P. C. VAN OORSCHOT(2007)] est une solution proposée en 2005 par l'université de Carleton dont le concept est de reprendre le meilleur de S-BGP et le meilleur de so-BGP afin de fournir une solution applicable immédiatement qui soit un compromis raisonnable entre la sécurité et la faisabilité.

PsBGP reprend l'infrastructure à clé publique de S-BGP : des autorités de confiance comme les RIR diffuseront des certificats qui permettront de réaliser l'authentification des numéros d'AS.

Pour savoir si un AS détient bien un préfixe, PsBGP considère qu'une infrastructure à clé publique est trop lourde à mettre en place et privilégie donc, comme SoBGP, un réseau de confiance : à intervalle régulier, chaque AS calcule une Prefix Assertion List, c'est-à-dire une liste des associations AS <-> préfixes et la diffuse à ses pairs. Chaque AS doit donc vérifier que les assertions qu'il connait correspondent bien à celles que ses voisins connaissent. Il est évidemment possible de configurer des poids afin de régler le niveau de confiance que l'on a en tel AS. Les contrôles concernant les chemins d'AS se font en suivant le même schéma acentré.

Avantages de PsBGP : il ne propose pas un nième modèle mais s'appuie sur deux solutions existantes (S-BGP et soBGP).

Inconvénients de PsBGP :

— Si un attaquant contrôle suffisamment d'AS (le papier original nous dit 2 AS), alors il peut émettre des PAL falsifiées et remettre en cause le modèle de PsBGP (donc, annoncer une route falsifiée). Néanmoins, cette attaque est clairement non triviale : pirater ou obtenir 2 numéros d'AS distincts ...

— À mon sens, PsBGP induit de la complexité : mettre en place une infrastructure à clé publique n'est pas simple (d'un point de vue opérationnel). Alors mélanger ça avec un réseau de confiance ...

— Problèmes habituels des réseaux de confiance : expiration des certificats ? Roulement des clés (en cas de perte, par exemple) ? Révocation ?

4.1.6 pgBGP

Pretty Good BGP (pgBGP) [Rexford(2006)] est une solution proposée en 2006 par l'université de New Mexico et dont le concept est de prendre le temps d'accorder sa confiance aux nouvelles routes. PgPGP utilise un modèle conservateur : quand plusieurs routes sont disponibles, il privilégie les anciennes, celles qui semblent fonctionner. PgBGP enraye donc la propagation de routes erronées.

Lorsqu'une nouvelle route apparaît (pour un préfixe connu, un nouveau préfixe ou un sous-préfixe), elle est considérée comme suspecte et n'est pas propagée. Au bout du délai de garde (24 heures, par défaut), si la route est encore annoncée, elle est considérée comme étant de confiance. PgBGP donne donc la possibilité aux opérateurs de réagir en levant des alertes.

Le fait d'imposer un délai à la propagation des routes peut paraître fatal à la résilience d'Internet mais pourtant, c'est bien cette dernière qui permet d'imposer un délai : plusieurs routes (au moins 2) sont généralement disponibles pour un même préfixe. De plus, le délai permet, par définition, d'éviter le problème de l'oscillation des routes.

Avantage : PgBGP n'utilisant pas la cryptographie, il n'y a donc pas d'infrastructure à déployer ni de lourdeur à craindre. Il n'y a pas non plus besoin de modifier le protocole BGP.

Inconvénients :

— Des faux positifs peuvent survenir : en cas de changement de connectivité (l'ancienne route, pourtant invalide, sera préférée durant le délai de garde), une connectivité "de secours" pourrait être considéré comme route principale/de confiance et le trafic envoyé dessus (alors que ce n'est pas la volonté de l'AS source).

— En l'absence de route alternative, la seule route disponible sera sélectionnée. Donc un attaquant peut forcer un retrait de la route par défaut grâce à une attaque par déni de service puis annoncer sa route malveillante. Elle sera sélectionnée d'office par le BGP classique et par PgBGP. Ce dernier n'apporte donc aucune sécurité supplémentaire dans ce scénario mais ne fait pas pire.

— PgBGP ne protège pas, dans son implémentation actuelle, contre un attaquant qui s'insère dans le chemin d'AS et qui est donc en position pour intercepter du trafic).

— PgBGP repose sur la vigilance des opérateurs : si une route passe le délai de garde, elle sera acceptée. Les grands opérateurs prendront-ils rapidement au sérieux les alertes PgBGP ?

4.1.7 Autres

De nombreuses autres méthodes visant à sécuriser le routage Internet apparaissent dans des papiers universitaires dont beaucoup sont restées à ce stade : MOAS Lists (basé sur la détection d'un conflit), Listen and Whisper (vérifier les routes en établissant des connexions TCP inter-routeurs et en utilisant de la cryptographie), Reachability Validation Graph (RVG, construction de graphes d'accessibilité), Inter-Domain Routing Validator (IRV, vérification des échanges BGP), ... Toutes ces solutions présentent des lacunes.

MOAS-List (2002) se base sur la détection d'un conflit. Quid des détournements de sous-préfixes ou de la corruption du chemin d'AS ? Quid des origines multiples légitimes (anycast, par exemple) ?

IRV (2003) propose que chaque AS crée un serveur IRV qui a autorité sur l'information de routage inter-domaine de son AS. Les serveurs IRV peuvent s'interroger mutuellement, sur une couche trans-

port sécurisée, afin vérifier et valider les messages BGP échangés. Selon moi, l'inconvénient majeur est qu'on recrée des IRR répartis qui ne seront pas mieux tenus à jour par les AS.

RVG (2004) propose de créer un graphe de validation d'accessibilité pour chaque préfixe. Les informations nécessaires à sa construction seraient mises à disposition par les opérateurs sur un système plus ou moins centralisé. L'échec des IRR nous montre qu'une telle solution (mise à disposition et mise à jour d'informations par les opérateurs de manière volontaire) est vouée à l'échec.

Listen and Whisper (2004) propose, d'une part, de surveiller si une route est réellement joignable en effectuant une connexion TCP de routeur à routeur, d'autre part d'utiliser une chaîne de condensats cryptographiques pour comparer plusieurs routes possibles entre elles et détecter de fausses routes. Listen est sensible au scan de port (tenter de maintenir une connexion TCP de manière fictive). Whisper ne peut que remonter une alerte sans dire l'origine du problème. Il est également sensible à des routeurs compromis qui injecteraient/supprimeraient des données.

4.2 Actuelles

4.2.1 ROVER

ROute Origin VERification (ROVER) [J. Gersch(2012)] est une alternative à RPKI+ROA présentée en mars 2012 au workshop de l'OARC et qui fait l'objet de brouillons à l'IETF. Elle a pour objectif de sécuriser uniquement l'origine, pas le chemin d'AS complet. ROVER part du principe que le titulaire d'un préfixe a aussi obtenu la délégation de la zone reverse (in-addr.arpa. / ip6.arpa.). Le principe est donc d'ajouter des enregistrements à cette zone reverse : RLOCK (Route LOCK) qui indique que l'on utilise ROVER pour valider les routes et que donc toute annonce faite par un AS n'apparaissant pas dans la zone reverse doit être rejetée. À cela s'ajoute un ou plusieurs enregistrements de type SRO (Secure Route Origin) qui indiqueront un AS autorisé à annoncer la route ainsi que, optionnellement, un transitaire autorisé. Il apparaît évident que DNSSEC est un prérequis au déploiement de ROVER. La validation de l'origine des préfixes ne sera pas faite sur les routeurs mais sur une machine auxiliaire afin de ne pas surcharger ces derniers.

ROVER est, certainement, un descendant de « DNS-based NLRI origin AS verification in BGP » (voir : `https://tools.ietf.org/html/draft-bates-bgp4-nlri-orig-verif-00`), méthode similaire à ROVER qui a fait l'objet d'un brouillon à l'IETF en 1998. Ce dernier proposait de créer une nouvelle hiérarchie : bgp.in-addr.arpa. Chaque sous-domaine indique quels AS est(sont) autorisés à être l'origine du préfixe et sur quelle longueur et/ou délègue la mise à disposition de cette information pour un sous-prefixe à un sous-domaine. bgp.in-addr.arpa. contient les informations que l'on retrouve actuellement dans le registre de l'IANA (exemple : 5/8 est délégué au RIPE).

L'avantage de ROVER est qu'il se base sur une infrastructure de confiance existante, ce qui évite d'avoir à en bâtir une nouvelle.

Les inconvénients sont :

— Il s'agit d'une solution supplémentaire qui s'appuie sur DNS. DNS, qui permet déjà de supprimer les centres des autres protocoles (mail, XMPP, ...), deviendrait ainsi un élément de plus en plus vital dans l'Internet, ce qui n'est pas sans danger.

— ROVER dépend du déploiement de DNSSEC, qui n'est qu'à son commencement.

— Il nécessite de changer le schéma de la délégation inversée pour les préfixes v4 qui ne tombent donc pas sur un octet (exemples : /19, /22, ...). Mais est-ce un mal étant donné la solution actuelle ?

La deuxième solution actuelle est RPKI+ROA, la seule solution normalisée dans les RFC 6480 à 6493, 6810 et 6811 (principalement).

4.3 Récapitulatif

Cette comparaison des différentes solutions est purement subjective et basée sur les critères communs que l'on peut dégager des différentes solutions. RPKI+ROA n'est pas comparée pour maintenir le suspens.

Critères :

— Facilité de déploiement. De 3 (meilleur) à 0 (pire). -1 si la solution repose sur la cryptographie (peu familière aux opérateurs), -1 si la solution combine plusieurs mécanismes (exemple : PKI et réseau de confiance).

— Efficacité. De 2 (meilleur) à 0 (pire) en fonction des vulnérabilités théoriques potentielles.

— Complète. De 2 (meilleur) à 0 (pire). -1 si la solution ne résout que l'un ou l'autre des problèmes (validation de l'origine OU validation du chemin).

— Modification de BGP : la solution modifie-t-elle BGP ?

— Normalisation. De 2 (meilleur) à 0 (pire). 0 : la solution ne fait l'objet d'aucun consensus et n'est pas déployée. 1 : la solution est déployée mais pas normalisée. 2 : la solution a fait l'objet d'une tentative de normalisation.

Critères	Alarmes	BCP	S-BGP	SoBGP	PsBGP	PgBGP	ROVER
Facilité de déploiement	***	***	**	**	*	***	**
Efficacité	**		**	**	*	*	**
Complète	**		**	**	**	**	*
Modification de BGP				Oui			
Normalisation	*	*	**	**			**

5 RPKI+ROA

Les documents du SIDR [WG(2012-2013)], le blog de Stéphane BORTZMEYER [BORTZMEYER(2012)] ainsi que sa présentation au FRnOG 19 [FRn(2012)] m'ont servi pour la rédaction de cette sous-partie.

5.1 Présentation succincte

RPKI+ROA (Resource Public Key Infrastructure + Route Origin Authorizations) est, depuis janvier 2013 [18], la solution normalisée à l'IETF pour initier la définition d'un routage inter-AS sécurisé. Il ne s'agit pas d'un nouveau protocole de routage dynamique mais d'une infrastructure couplée à un ensemble d'objets cryptographiques et de procédures. Le tout forme un ensemble complémentaire au routage dynamique traditionnel permettant de construire, autour de ce dernier, un mécanisme de sécurisation. RPKI+ROA est indépendante du protocole de routage dynamique utilisé : elle ne casse pas le protocole actuel, BGP [19] et fonctionnerait également avec un éventuel nouveau protocole.

L'objectif de RPKI+ROA est d'empêcher l'usurpation de préfixes. De ce fait, RPKI+ROA se focalise sur la sécurisation de l'origine c'est-à-dire tente d'apporter une réponse sécurisée à une question comme : « cet AS a-t-il l'autorisation de s'annoncer comme étant l'origine de tel préfixe ? ». De ce fait, elle n'est que la première étape du chemin qui conduit à la sécurisation intégrale (origine + chemin d'AS) du routage inter-AS. Cette dernière, bien qu'étant aussi en réflexion au sein de l'IETF, relève d'une complexité grandement supérieure.

Pour atteindre son objectif, RPKI+ROA a recours à toute la cryptographie asymétrique habituelle (paire de clés, certificats x509, ...) afin d'authentifier une partie des messages BGP : l'origine. Il n'y a aucune notion de confidentialité dans RPKI+ROA car les données du routage inter-AS sont, par définition, publiques. Il n'y a pas non plus de notion de sécurisation du canal car, comme je l'ai déjà dit, plusieurs solutions permettent déjà de sécuriser la communication entre deux pairs et, surtout, sécuriser le canal ne participe pas à l'objectif de RPKI+ROA.

RPKI+ROA est évolutive car elle ne dépend pas d'un algorithme cryptographique particulier : ce n'est qu'un paramètre. Si demain les progrès de la cryptanalyse permettent de casser les algorithmes utilisés actuellement (RSA 2048 bits et SHA256), il suffira d'en changer. La gêne sera donc temporaire et ne nécessitera pas de revoir entièrement RPKI+ROA.

18. Les premiers RFC décrivant RPKI+ROA ont été publiés en février 2012 mais, en l'absence de la définition d'un composant clé de l'infrastructure, il m'apparaît impossible de parler de normalisation à ce stade. Depuis janvier 2013, toute l'infrastructure est décrite. Selon moi, il reste quelques points à préciser mais ils ne remettront pas fondamentalement en cause l'infrastructure. C'est pourquoi je présente RPKI+ROA comme une solution normalisée dès lors que l'on prend en compte les RFC de janvier 2013.

19. Une partie des objets de RPKI+ROA pourraient être distribuée dans un nouvel attribut BGP d'un message BGP UPDATE, tout au plus.

5.2 Présentation détaillée

5.2.1 Vue globale

L'architecture générale de RPKI+ROA est présentée dans l'illustration 3.

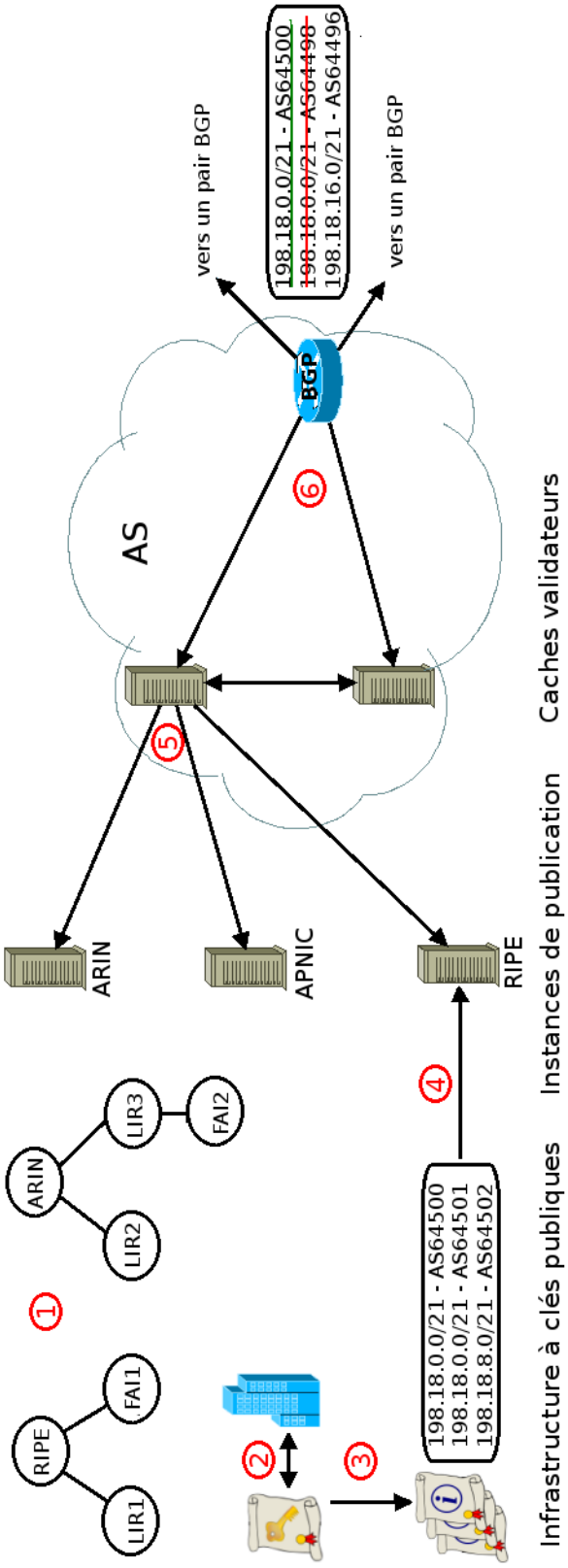

FIGURE 3 – Architecture RPKI+ROA. Illustrations : https ://commons.wikimedia.org/wiki/File :Public-Key-Infrastructure.svg

Légende :

1. On met en place une nouvelle infrastructure à clés publiques (PKI) arborescente à laquelle participent toutes les organisations qui oeuvrent déjà à l'allocation des ressources numériques uniques d'Internet (préfixes IP et ASN). Les Autorités de Certification x509 PKIX dites "classiques" (DigiCert, VeriSign, Thawte, ...) n'ont aucun rôle dans cette nouvelle PKI.

   Notons que, même si c'est l'IANA qui est au sommet de la distribution des ressources numériques sur Internet, personne n'a souhaité que cet organisme soit l'unique racine de notre infrastructure à clés publiques. Actuellement, chaque RIR est une racine de notre nouvelle PKI et délègue aux LIR, qui délèguent ...

   Cette infrastructure à clés publiques est nommée Resource Public Key Infrastructure (RPKI).

2. Les organisations qui allouent les ressources uniques d'Internet distribuent des certificats x509 à tous les titulaires de ressources. Ces certificats permettent à un titulaire de prouver qu'il est bien le titulaire d'une ou de plusieurs ressources : « le FAI Example est titulaire de 192.0.2.0/24 ».

3. Les certificats permettent de signer différents objets cryptographiques, au format Cryptographic Message Syntax, dont les plus importants sont les ROA (Route Origin Authorizations) : un ROA donne l'autorisation à un AS de s'annoncer comme étant l'origine de tout ou partie des préfixes du signataire : « L'AS 64501 est autorisé à annoncer les préfixes 192.0.2.0/24 et 2001 :0DB8 : :/32 ».

4. Les certificats et les objets sont distribués publiquement à travers un système de dépôts possiblement distribué qui n'offre aucune sécurité. Un dépôt est nommé « instance de publication ». À minima, tous les contenus de tous les dépôts sont récupérables avec rsync. Actuellement, chaque RIR héberge une instance de publication.

5. Les opérateurs mettent en place des caches validateurs qui sont chargés de récupérer tous les objets et certificats présents dans le système de dépôts et d'effectuer les calculs cryptographiques permettant d'affirmer que les certificats et objets n'ont pas été altérés et qu'ils sont valides (syntaxe, sémantique). N'importe qui peut installer un cache validateur mais il est conseillé aux opérateurs de mettre en place une infrastructure en strates (NTP-like) pour ne pas surcharger le système de dépôts.

6. Régulièrement, les routeurs de bordure d'un AS communiquent avec un ou plusieurs caches validateurs afin d'obtenir la liste des autorisations (contenu des ROA) valides. Cette communication se fait selon un nouveau protocole basé sur TCP : RPKI To Router (RTR). Le routeur stocke les autorisations reçues et s'en sert, à la reception d'une annonce BGP, comme d'un paramètre supplémentaire pour choisir les routes et construire sa table de routage. Chaque

opérateur réseau peut choisir différentes actions en fonction de l'état de l'origine d'une route : refuser les annonces invalides, leur affecter une préférence locale plus faible, ...

Comme RPKI+ROA est composée d'un ensemble d'éléments, je vais présenter, progressivement, tous ces composants de maniètre très détaillée afin de produire une description plus précise de RPKI+ROA.

5.2.2 Resource Public Key Infrastructure

RPKI est une infrastructure à clés publiques, c'est-à-dire, des procédures humaines et techniques permettant de gérer le cycle de vie (créer, valider, distribuer, révoquer, ..) des certificats numériques.

Dans le cas de la RPKI, il s'agit de distribuer un certificat, nommé « CA Certificate » (soit certificat d'autorité de certification en français) ou « Resources Certificate », à toutes les organisations qui participent à la distribution des ressources sur Internet (RIR, LIR, éventuellement opérateur réseau) et aux titulaires d'au moins un préfixe IP (opérateurs réseau) afin que ces derniers prouvent qu'ils sont bien les titulaires de ces préfixes. La création et la distribution des certificats d'AC seront réalisées par les mêmes organisations qui sont déjà chargées de distribuer les ressources uniques sur Internet : RIR -> LIR.

Ainsi :

- Les titulaires d'au moins un préfixe Provider-Independent reçoivent un certificat AC.
- Un opérateur réseau disposant d'au moins un préfixe Provider-Aggregatable obtenu auprès d'un opérateur/LIR et qui dispose d'un seul transitaire, c'est-à-dire l'opérateur/LIR qui lui fournit le préfixe lui-même, ne fait pas partie de la RPKI : le LIR est le titulaire du préfixe, c'est donc lui qui recevra un certificat d'AC et devra générer les objets cryptographiques nécessaires.
- Un opérateur réseau disposant d'au moins un préfixe PA obtenu auprès d'un opérateur/LIR et qui dispose de plusieurs transitaires est le cas le plus compliqué de la RPKI. Cette configuration, appelée « multi-homing », peut être traitée de plusieurs façons dans la RPKI :
  - La solution conseillée est que ce type d'opérateurs dispose de son propre ASN et annonce le préfixe "alloué". Dans ce cas, l'opérateur/LIR peut distribuer un certificat d'AC à cet opérateur afin que ce dernier génère lui-même ses objets cryptographiques. L'opérateur/LIR peut aussi ne pas distribuer de certificat et générer lui-même les objets cryptographiques.
  - La solution déconseillée est que ce type d'opérateurs ne dispose pas de son propre ASN et n'annonce pas le préfixe "alloué". Dans ce cas, l'opérateur/LIR générera les objets cryptographiques pour chacun des transitaires, lui compris.

Notons que, bien que RPKI+ROA a pour objectif premier la sécurisation du routage, il est évident que la distribution de certificats aux titulaires de préfixes servira à d'autres usages qui nécessitent de

savoir si un interlocuteur est bien le titulaire d'une ressource. Exemple d'actualité : la revente de blocs IPv4.

5.2.2.1 Format des certificats Les certificats de la RPKI sont des certificats x509. Néanmoins, le format des certificats x509 tel que défini par l'UIT est d'une telle complexité que le format n'a jamais été utilisé intégralement : l'usage se fait en suivant des profils c'est-à-dire une restriction des usages possibles. Le profil le plus utilisé est PKIX. Il est défini dans le RFC 5280 (et ses prédécesseurs) et il indique quelles parties de la norme x509 sont utilisées sur Internet (TLS).

Les certificats de la RPKI suivent un nouveau profil, « X.509 PKIX Resource Certificates », dérivé de PKIX et défini dans le RFC 6487, qui fixe le format et la sémantique des certificats dédiés à la certification d'un droit d'utilisation sur les ressources uniques d'Internet.

Ce profil ajoute des contraintes supplémentaires par rapport au profil PKIX dont les principales sont :

— Des champs prennent des valeurs obligatoires. La version du certificat doit être 3, l'algorithme de signature doit actuellement être RSA-SHA256. Le sujet est fixé par l'AC, doit être unique au sein d'une AC et peut être une valeur aléatoire, ...

— Les extensions du RFC 3779, qui permettent de décrire les ressources Internet sont obligatoires. Les autres extensions sont interdites : on construit un profil dédié à la certification d'un droit d'utilisation sur les ressources Internet, pas un profil multi-usages.

— La validation se fait toujours en vérifiant les champs du certificat, sa validité (non révoqué, période de validité en cours) et en essayant de construire une chaîne de confiance jusqu'à un certificat racine mais il faut aussi vérifier la cohérence des ressources annoncées entre un certificat "enfant" et son "parent" : le certificat parent doit inclure ou englober les ressources annoncées dans le certificat "enfant".

— On ne conserve ni l'infrastructure de distribution ni le modèle de sécurité de x509. D'une part, comme nous venons de le voir, les Autorités de Certification x509 "classiques" (DigiCet, VeriSign, Thawte, ...) n'ont aucun rôle dans la RPKI. D'autre part, jusque-là, x509 apportait une certitude sur l'identité de la personne ou de l'organisation inscrite dans le champ "Subject" d'un certificat : cette personne/organisation est bien ce qu'elle prétend être. Dans le cas de la RPKI, on ignore le champ "Subject", on ne cherche pas à prouver l'identité d'une personne ou d'une organisation mais à faire une association entre l'organisation qui détient la clé privée associée au certificat et une ou plusieurs ressources Internet : celui qui détient la clé privée est le titulaire de la ressource (et peut en redistribuer une partie). Bien sûr, il ne suffit pas de voler une clé privée pour s'emparer d'une ressources : tout comme les noms de domaine ou d'autres secteurs, des mécanismes de récupération basés sur les informations fournies au RIR/LIR sont prévus si votre clé privée est compromise.

5.2.3 Des objets cryptographiques

Actuellement, deux types d'objets cryptographiques sont normalisés. Rien n'empêche la normalisation de types d'objets supplémentaires dans le futur.

5.2.3.1 ROA Le premier type d'objets est ROA (Route Origin Authorizations). Ce sont les objets centraux de RPKI+ROA. Chaque ROA exprime, pour un AS, une autorisation d'annoncer au moins un préfixe. Un ROA contient une ou plusieurs assertions comme : « L'AS 64501 est autorisé à annoncer les préfixes 192.0.2.0/24 et 2001 :0DB8 : :/32 ». Un ROA permet de lister plusieurs préfixes autorisés à être annoncés par un AS mais il ne permet pas d'autoriser un ou plusieurs préfixes à être annoncés par plusieurs AS. Exemple impossible : « Les AS 64501 et 64502 sont autorisés à annoncer les préfixes 192.0.2.0/24 et 2001 :0DB8 : :/32 ». Dans ces cas-là (PI et PA multi-homée), il faudra générer plusieurs ROA, un par AS que l'on souhaite autoriser, chacun d'entre eux listant les deux préfixes autorisés.

5.2.3.1.1 Taille et topologies Par défaut, l'autorisation contenue dans un ROA est stricte : seule une annonce dont le préfixe a strictement la même longueur sera acceptée. Avec les exemples précédents, une annonce plus précise portant, par exemple, sur 192.0.2.0/25 sera refusée. Cependant, il est possible d'annoncer plusieurs longueurs pour un même préfixe sans pour autant multiplier les ROA grâce à une propriété, « MaxLenght », associée à chaque préfixe dans un ROA. Grâce à cette propriété, on peut exprimer des assertions comme :

- « 192.0.2.0/24 longueur maximale /24 » : L'AS pourra seulement annoncer 192.0.2.0/24. Les annonces pour les préfixes plus précis (/25, ...) seront refusés. C'est le comportement par défaut si « MaxLenght » n'est pas spécifiée.
- « 192.0.2.0/24 longueur maximale /26 » : L'AS pourra annoncer 192.0.2.0/24 ou n'importe quel préfixe plus spécifique ayant une longueur inférieure ou égale à /26 : les deux /25 recouverts par le /24 ou même les quatre /26 englobés par le /24. En revanche, il ne pourra pas annoncer un /27.
- « 192.0.2.0/24 longueur maximale /26 192.0.2.0/28 » : L'AS pourra annoncer 192.0.2.0/24 ou n'importe quel préfixe plus spécifique ayant une longueur inférieure ou égale à /26 ainsi que 192.0.2.0/28.

Il est évidemment possible de découper les autorisations entre plusieurs AS (mais dans plusieurs ROA, pour rappel) : le titulaire d'un préfixe /24 peut très bien autoriser un AS à annoncer uniquement le /24 alors qu'il permet à un deuxième AS d'annoncer le même préfixe de /24 jusqu'à /26 et à un troisième AS d'annoncer un /28. Seule la cohérence du découpage est bloquante.

Les ROA s'adaptent aux topologies réelles. Pour avoir un aperçu des possibilités, on peut parcourir la section 3 du RFC 6907 qui décrit plusieurs scénarios et les objets à créer dans ces situations.

5.2.3.1.2 Interdire l'annonce d'un préfixe Pour interdire l'annonce d'un ou de plusieurs préfixes, il suffit de créer un ROA qui les associe à l'ASN 0, qui est un numéro d'AS réservé c'est-à-dire un numéro qui ne doit jamais apparaître dans une annonce BGP. Ainsi, si une annonce contient un préfixe interdit, elle sera refusée car l'AS de l'annonce (exemple : 64501) ne conviendra pas à celui indiqué dans le ROA (AS 0). Si une annonce contient le même préfixe mais associé à l'AS 0, l'annonce sera acceptée d'un point de vue du ROA mais refusée car l'ASN 0 est réservé.

Cette méthode couvre au moins trois usages :

— Interdire l'annonce des préfixes réservés (privés, locaux, documentation, multicast local, ...) ou non alloués. Normalement, si l'on suit les BCP, ces préfixes devraient être filtrés et ne devraient jamais circuler, même partiellement sur Internet. Or, nous avons vu des contre-exemples. C'est l'IANA qui est chargé d'émettre les certificats et les objets pour ces préfixes. Attention : cela ne remet pas en cause le fait qu'en pratique personne n'a voulu l'IANA comme seule racine de la RPKI : l'IANA ne sera ici qu'une racine supplémentaire de la RPKI.

— Interdire l'annonce des préfixes alloués mais pas encore annoncés. Un opérateur peut avoir fait une demande de préfixe en prévision d'une augmentation de ses capacités mais ne pas encore utiliser la nouvelle allocation. Un nouvel opérateur peut avoir obtenu une allocation sans être encore prêt à l'annoncer car il n'a pas encore déployé son infrastructure. Exemple réel : ARN, FAI associatif alsacien. Il s'est écoulé un trimestre entre l'obtention des ressources (préfixes v4/v6 et ASN) et la mise en place de l'infrastructure.

— Les titulaires de préfixes alloués mais pas annoncés globalement peuvent émettre des ROA pour être sûrs que leurs préfixes ne seront pas annoncés globalement. Le débat revient régulièrement sur les listes de discussion orientées réseaux : « un préfixe alloué mais non annoncé globalement sur Internet est-il un préfixe non utilisé ? ». Pourtant, des usages, par ailleurs reconnus par les RIR [20], peuvent nécessiter un adressage public non annoncé :

  — Une organisation a pu obtenir des préfixes avant l'apparition du RFC 1918 en 1996, avoir numéroté son réseau et ne pas vouloir supporter le coût d'un re-numérotage compatible RFC 1918.

  — Des usages proscrivent le NA(P)T (plusieurs serveurs) et des protocoles fonctionnent mais avec des bricolages (ICMP, FTP, SIP, ...), d'où des besoins d'un adressage "privé".

  — Des organisations peuvent faire des interconnexions privées entre elles en utilisant un préfixe public. BGP permet même cela avec une communauté spéciale : NO-EXPORT. Cette démarche se justifie d'autant plus lorsque le besoin d'un adressage unique globalement (ce que ne garantit pas l'adressage RFC 1918) se fait sentir.

20. Voir, par exemple, la politique d'allocation de l'ARIN : `https://www.arin.net/policy/nrpm.html#four35`.

5.2.3.2 Ghostbusters

Le deuxième type d'objet est Ghostbusters. Ce sont des objets annexes de RPKI+ROA mais ils prendront leur importance lors des débogages. En effet, tous les déploiements de services et plus particulièrement de systèmes de sécurité (DNSSEC, par exemple) nous ont montrés qu'il y a toujours des problèmes et qu'il faut donc être en mesure de pourvoir faire remonter des informations jusqu'aux personnes qui sont en position d'agir.

Dans le cadre de RPKI+ROA, de nombreux problèmes peuvent survenir : certificat expiré, un ROA ne correspond plus avec les annonces légitimes, ...

Les personnes à même d'agir seront souvent celles qui disposent des clés privées des certificats d'AC. Comment les contacter ?

— Pour rappel, les certificats de la RPKI ne contiennent pas le nom de leur titulaire (le champ « Subject » peut être généré aléatoirement par l'AC qui a signé le certificat).

— Les bases de données des RIR, utilisées par whois, ne sont pas forcement à jour. De plus, rien ne garantit que la personne qui a fait la demande des ressources est aussi celle qui s'occupe de RPKI+ROA au sein d'une grande organisation.

Bien que Twitter ait fait ses preuves comme outil de gestion de crise, les auteurs de RPKI+ROA ont imaginé une solution plus pérenne et indépendante. Au moins une des personnes responsables de la gestion des certificats d'AC et donc de la RPKI+ROA au sein d'une organisation peut, si elle le souhaite (la démarche est volontaire), créer un objet Ghostbusters contenant une vCard simplifiée : elle ne peut contenir qu'une seule fois les propriétés VERSION :4.0, FN [nom/prénom], ORG [organisation] et au moins une fois les propriétés ADR [adresse postale], TEL et EMAIL.

Les Ghostbusters sont des objets destinés aux humains, pas aux machines : leur contenu ne sera pas utilisé par les routeurs. Il est évident que les informations fournies dans un Ghostbusters sont publiques comme tous les objets de RPKI+ROA, qu'elles peuvent ne pas être fiables (comme un whois) et que la personne désignée ne devient pas titulaire des ressources de l'AC associée au Ghostbusters.

Les objets de type Ghostbusters créent donc un deuxième canal de communication, parallèle à whois, qui augmente la probabilité de pouvoir contacter rapidement la personne responsable de RPKI+ROA dans une organisation qui sera à même de corriger le dysfonctionnement.

5.2.3.3 Émission et signature des objets cryptographiques

Tous les objets cryptographiques de RPKI+ROA sont émis par le titulaire d'un préfixe qui utilisera la clé privée associée au certificat d'AC qu'il a obtenu pour les signer. Plus précisément, le titulaire d'un certificat d'AC générera un certificat dit End-Entity Certificat (EE) pour chaque objet (ROA, Ghostbusters, ...) qu'il souhaite signer et signera ce certificat EE avec la partie privée qui correspond à la partie publique contenue dans le certificat d'AC.

Un certificat EE se différencie d'un certificat d'AC par le fait qu'il ne peut pas déléguer de ressources, c'est-à-dire que la partie privée qui y est associée ne peut pas servir à signer d'autres certificats

d'AC : la partie privée associée à un EE peut uniquement signer les objets de RPKI+ROA. La présence de la contrainte « AC» au sein d'un certificat permet d'indiquer si ce dernier est un certificat d'AC ou un certificat EE.

Un objet est signé par un seul EE et un EE doit signer un seul objet. Cette association unique présente des avantages au niveau de la sécurité et de la flexibilité :

— On peut détruire la clé privée qui correspond à un certificat EE après signature. La seule information cruciale à protéger, au sens intégrité et disponibilité, est la clé privée associée au certificat d'AC. Plus on réduit le matériel à protéger, plus on augmente la probabilité que la protection soit efficace.

— La révocation d'un seul objet est facile : il suffit de révoquer le certificat EE auquel il est associé.

La question que l'on peut se poser est « Pourquoi ne pas tout signer (certificats d'AC des AC "filles" ainsi que tous les objets) avec le certificat d'AC ? ». Les deux raisons principales que je vois sont la flexibilité-simplicité et la sécurité. On peut illustrer la sécurité en disant qu'avec le mécanisme certificat d'AC + certificat EE, le matériel cryptographique est partagé entre plusieurs paires de clés (dont certaines peuvent être détruites, pour rappel) : l'une signe les certificats d'AC, les autres signent chaque objet de RPKI+ROA. Cela force également le changement des paires de clés qui signent les objets. On peut illustrer la flexibilité et la simplicité avec un problème comme celui-ci : comment révoquer un ROA (car il ne correspond plus à la réalité des annonces) sans pour autant révoquer l'intégralité des autres ROA encore valides si l'on n'a qu'un certificat d'AC pour tout signer ? Il faudrait créer un mécanisme de révocation supplémentaire qui rendrait RPKI+ROA plus complexe.

5.2.3.4 Format des objets cryptographiques Cryptographic Message Syntax (CMS) est un format générique d'encapsulation de données vouées à être protégées par de la cryptographie. CMS est normalisé dans le RFC 5652 et est utilisé dans quelques applications, la plus importante, jusqu'à présent, étant S/MIME.

Tous les objets de la RPKI (ROA, Ghostbusters, ...) utilisent un gabarit CMS normalisé dans le RFC 6488. L'objectif de ce gabarit CMS est le même que celui du profil x509 pour les certificats de la RPKI ou que le profil vCard pour les objets Ghostbusters : adapter la norme, ici CMS, à l'usage considéré. Ce gabarit fixe les propriétés utilisables ou non, restreint les valeurs possibles de ces propriétés (exemple : les algorithmes cryptographiques utilisables) et défini la procédure de vérification/validation commune à tous les objets de la RPKI.

Le gabarit normalise la structure générale commune à tous les objets relatifs à la RPKI. Le contenu, l'information que l'on cherche à signer, varie d'un type d'objet à un autre : dans un ROA le contenu est un ASN et une liste de préfixes IP alors que dans un Ghostbusters, il s'agit d'une vCard minimaliste. Il faut donc normaliser chaque classe d'objets afin de définir les propriétés et des règles de validation plus spécifiques propres à la classe d'objets.

5.2.3.5 Schéma de nommage des objets cryptographiques

Tous les objets de RPKI+ROA suivent un schéma de nommage recommandé : le nom est dérivé de la clé publique associée à l'objet : sha1 + base64url du RFC 4648 à quoi s'ajoute un suffixe normalisé dépendant de la classe de l'objet (.cert pour les certificats d'AC, .roa pour les ROA, .mft pour les Manifest, ...). Un certificat d'AC aura donc un nom dérivé de sa clé publique. Les autres objets (ROA, Ghostbusters, ...) auront un nom dérivé de la clé publique de leur certificat EE.

Ce schéma de nommage permet de conserver un nom persistant pour les objets ainsi qu'une chaîne de validation tant que les paires de clés ne sont pas changées. Ce schéma de nommage est plus important pour les certificats d'AC que pour les autres objets car il amène de la flexibilité. Par exemple : un LIR reçoit de nouvelles ressources de son RIR : son certificat d'AC doit être mis à jour pour refléter ces nouvelles allocations. Sans ce schéma de nommage, le LIR devrait possiblement resigner tous ses objets, y compris les certificats des AC filles (car les certificats ont liés entre eux par l'équivalent de pointeurs, nous y reviendrons).

Ce schéma de nommage apporte aussi la garantie de sécurité que certains anciens objets seront bien écrasés par leur nouvelle version.

5.2.4 Des points de publication

La question qui se pose désormais est de savoir comment distribuer tous les objets de RPKI+ROA (au sens large : certificats d'AC, ROA, Ghostbusters, ...) aux caches validateurs afin que ces derniers puissent, ou non, valider les ROA et transmettre, le cas échéant, les autorisations qu'ils contiennent aux routeurs.

RPKI+ROA utilise un système de dépôts.

5.2.4.1 Structure du système de dépôts

À chaque certificat d'AC est associé un dossier (au sens dossier sur un système de fichiers) contenant tous les objets non expirés signés avec la clé privée associée à ce certificat (ROA, autre certificat d'AC, ...). Ce dossier est appelé un point de publication, il est complet, c'est-à-dire qu'il contient, en permanence, tous les objets non expirés d'une AC. Un point de publication est l'endroit de référence pour récupérer les objets d'une AC [21].

Un ou plusieurs points de publication peuvent être diffusés par une même grappe de serveurs, par une même entité et donc partager un début d'URI commun. Concrètement, un ou plusieurs dossiers sont mis à disposition par une même entité. Cet ensemble est appelé une instance de publication. Les points de publication contenus dans une instance n'ont pas forcement de liens entre eux bien que ce soit le cas dans la pratique : les points de publication sont regroupés car une AC est membre ou cliente d'une autre.

21. En réalité, plusieurs AC peuvent partager un même point de publication mais on entre là dans un cas d'utilisation qui sera rare en pratique.

Une seule instance n'a pas vocation à contenir tous les points de publication et donc tous les objets de RPKI+ROA. C'est d'ailleurs ce qui se passe en pratique : les RIR mettent à disposition une instance qui contient leur point de publication ainsi que ceux de leurs membres (les LIR et les titulaires de PI, pour rappel). L'instance de chaque RIR n'est pas complète : elle ne contient pas les points de publication des autres RIR. De même, un LIR ou un opérateur pourrait héberger sa propre instance contenant uniquement son point de publication.

5.2.4.2 Une hiérarchie distribuée Le système de dépôts de RPKI+ROA forme une hiérarchie distribuée un peu à la manière de l'arbre inversé que forme le DNS (à l'exception bien sûr qu'il n'y a qu'une seule racine globale reconnue au sommet du DNS, celle de l'ICANN (on ne s'attardera pas sur les projets de racines alternatives) alors qu'il y a, à l'heure actuelle, cinq racines dans RPKI+ROA). Un point de publication "fait autorité" pour une AC.

On peut alors se demander « sachant qu'un point de publication est complet mais qu'une instance peut-être incomplète, comment un cache-validateur peut être sûr de récupérer tous les objets de RPKI+ROA afin de permettre à un router de prendre les meilleures décisions possibles ? ».

Chaque certificat doit contenir, entre autres, deux propriétés :

— Authority Information Access (AIA) : indique, sous forme d'URI quel est le certificat d'AC qui a signé ce certificat. Ce champ est anecdotique dans cette explication.

— Subject Information Access (SIA) : indique, uniquement pour les certificats d'AC, l'URI du point de publication de l'AC.

Ainsi, le champ SIA permet de mailler un certificat d'AC à son point de publication. Exemple en image :

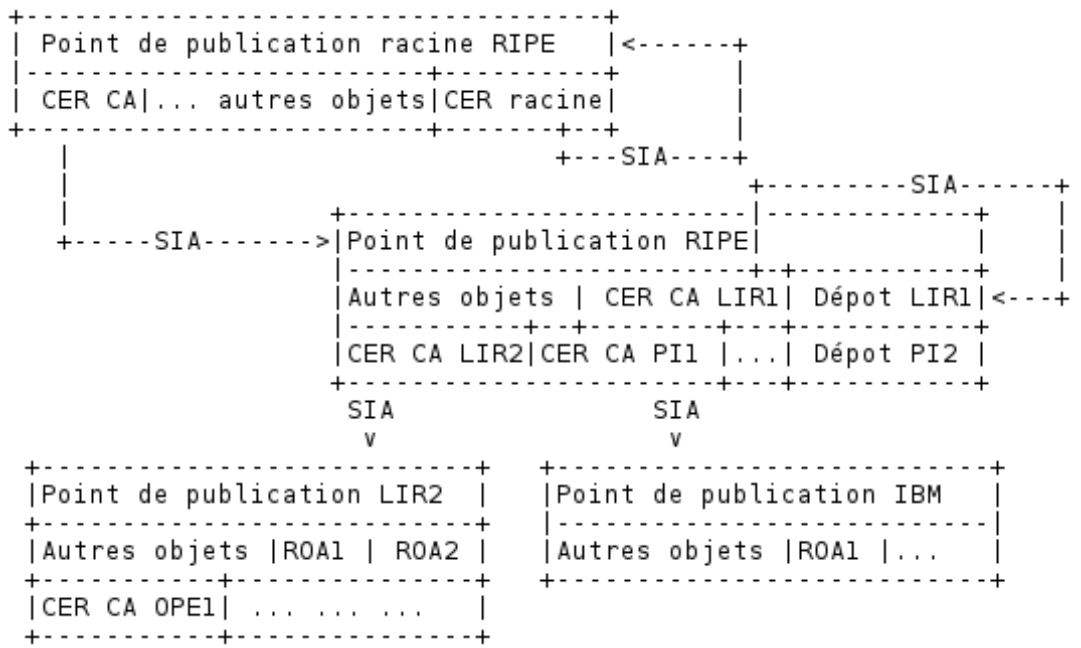

FIGURE 4 – Explication du chaînage des certificats dans RPKI+ROA.

Le premier point de publication est celui de la racine du RIPE. Le certificat racine du RIPE, contenu dans ce point de publication, est auto-signé et son SIA pointe sur le point de publication lui-même. Ce certificat signe un deuxième certificat, appartenant toujours au RIPE [22]. Ce deuxième certificat sert à signer tous les certificats des membres du RIPE. Son SIA désigne donc un deuxième point de publication. Dans notre exemple, LIR2 et IBM France (qui a, on le supposera, des blocs PI obtenus auprès du RIPE) ont choisi d'héberger leur point de publication en dehors de l'instance du RIPE. Les champs SIA de leurs certificats respectifs pointent donc vers leurs points de publication respectifs. Le LIR1 et le membre PI2 ont choisi de faire héberger leur point de publication par le RIPE. Les champs SIA de leurs certificats pointent donc sur un sous-dossier de l'instance de publication du RIPE.

Dès lors, pour qu'un cache validateur soit sûr de récupérer tous les objets de RPKI+ROA, il suffit de suivre les SIA des certificats d'AC en amorçant cet algorithme avec les cinq certificats racines actuels. Chaque SIA conduit à un point de publication dont il faut récupèrer tout le contenu. Si ce point de publication contient des certificats d'AC, on suit récursivement les SIA. Au final, on aura parcouru tous les points de publication même si chacun d'entre eux se trouvait dans une instance de publication différente stockée sur différents serveurs hébergés par des opérateurs différents.

Bien que, dans la théorie, le système de dépôts de RPKI+ROA puisse être totalement distribué, dans la pratique, et à mon avis, les seules instances seront celles des RIR et de, peut-être, quelques gros LIR.

Chaque point de publication doit être accessible avec rsync. D'autres protocoles d'accès supplémentaires peuvent être mis en place selon les volontés locales mais ce protocole doit offrir des fonctionnalités de base : permettre un téléchargement complet ou partiel (incrémental) des objets ainsi que la modification d'un point de publication, sous condition d'authentification et d'autorisation, afin qu'une AC qui délègue la gestion de son point de publication à une entité (exemple : à un RIR) puisse mettre à jour son point de publication publié dans l'instance de cette entité.

5.2.4.3 Sécurité des points de publication

5.2.4.3.1 Disponibilité Le système de dépôts est crucial pour la RPKI : si un point de publication n'est pas accessible, s'il manque des objets, ... la décision de privilégier une route par rapport à une autre peut-être faussée puisqu'on n'a pas la bonne information.

22. Cette itération supplémentaire amène de la flexibilité et de la sécurité : le certificat racine, garant de la sécurité de la chaîne, est conservé en sécurité pendant que le deuxième certificat sert à signer les certificats des membres. De plus, il est plus facile de changer le certificat intermédiaire que le certificat racine qui sera récupéré et stocké par les caches validateurs.

Pour assurer la disponibilité d'un point de publication (ou d'une instance), l'un des critères de la sécurité, il n'existe pas de méthode magique : le point de publication (ou l'instance) doit être servi par une infrastructure redondée qui assure sa haute disponibilité.

Notons que le fait que chaque AC puisse héberger son point de publication où bon lui semble n'est pas une réponse complète à la problèmatique de disponibilité : la RPKI ne bénéficie pas du même effet entonnoir que le DNS. Dans le DNS, la racine est beaucoup plus sollicitée que le TLD fr. qui est lui-même plus sollicité que u-strasbg.fr. Il y a donc une différence de charge entre les serveurs qui font autorité sur la racine et ceux qui font autorité sur u-strasbg.fr. Dans la RPKI, tous les opérateurs qui font de la validation doivent récupérer tous les objets. Chaque point de publication doit donc supporter la même charge.

5.2.4.3.2 Authenticité et intégrité Les instances de publication n'offrent aucune garantie de sécurité, ni dans l'hébergement des données ni lors de la récupération de leur contenu : l'authenticité et l'intégrité des objets cryptographiques seront vérifiées à réception par les caches validateurs.

La chaîne de validation, qui va d'un certificat racine jusqu'au certificat EE associé à un objet en passant par tous les certificats d'AC intermédiaires permet de protéger les objets (ROA, Ghostbusters, ...) contre les altérations et contre les ajouts d'objets : il faut les clés privées pour générer des signatures valides et pour ne pas casser la chaîne de validation.

En revanche, les signatures ne peuvent pas protéger contre la suppression malveillante d'un objet d'un point de publication ou la substitution d'un objet par son ancienne version non expirée et non révoquée (car, dans ce dernier cas, la chaîne de validation est encore valide). Notons que ces attaques peuvent survenir dans le système de dépôts lui-même ou durant la récupération des objets par les usagers finaux car les RFC ne prévoient pas un mécanisme de sécurisation du canal donc une attaque de l'homme du milieu est possible.

Pour contrecarrer ces risques, on introduit un nouvel objet : les Manifest. Un Manifest est un objet signé qui contient une liste des sommes de contrôle (nom et condensat sha256) de tous les objets (certificats d'AC, ROA, Ghostbusters, ...) non expirés et non révoqués contenus dans un point de publication. C'est lui qui garantit qu'un point de publication est récupéré dans son intégralité. Comme tous les autres objets de la RPKI, le Manifest est un objet CMS, signé par un certificat EE, qui est distribué via le système de dépôts. Ce type d'objet suit le schéma de nommage recommandé et porte le suffixe normalisé « mft ». Il n'y a qu'un seul Manifest par AC à un instant du temps.

Un Manifest a un intervalle de validité (souvent 24 heures mais ce n'est pas normalisé). Il doit donc être re-généré régulièrement, soit car il va expirer, soit car le contenu du point de publication a changé : objet ajouté, supprimé, modifié et le Manifest doit correspondre au contenu.

Petite subtilité : la clé privée associée à un certificat EE peut servir à signer une séquence de Manifest. En effet, un Manifest étant unique par AC et par instant de temps, la double association

« un certificat EE signe un unique objet et un objet est signé par un unique certificat EE » reste vraie. Ce raisonnement "en séquence" est aussi valable pour les autres objets de RPKI+ROA mais il ne se justifie pas car un Manifest doit être re-générer souvent, ce qui n'est pas le cas des autres objets : un ROA ne change ni n'expire pas tous les jours, ça n'a donc aucun sens de réutiliser la même paire de clés. Or, générer des paires de clés imprévisibles sur une courte échelle de temps, comme c'est nécessaire pour les Manifest, peut être vu comme une contrainte supplémentaire. Néanmoins, il est évident que la réutilisation d'une paire de clés entraîne une augmentation du risque de substitution malveillante d'un Manifest par sa précédente version.

Lors de la récupération du système de dépôts par les caches validateurs, plusieurs situations peuvent survenir : il n'y a pas de Manifest dans un point de publication, le Manifest a expiré, il n'y a pas une totale correspondance entre la liste des fichiers contenue dans le Manifest et le contenu réel du point de publication et/ou une somme de contrôle ne correspond pas. Chacune de ces situations peut indiquer une erreur (suppression accidentelle du Manifest, oubli de re-générer le Manifest, ...) ou une attaque. Au début du déploiement de RPKI+ROA, la probabilité qu'il s'agisse d'une erreur et non d'une attaque est élevée. Chacune de ces situations présente un risque différent car certaines des attaques peuvent être confirmées par d'autres techniques alors que la détection d'autres attaques repose totalement sur le Manifest. Le RFC laisse le choix à l'usager final de refuser ou non les objets d'un point de publication dans le cas où ces situations se produisent.

5.2.5 Des caches validateurs

Les routeurs de la DFZ n'interrogent pas directement le système de dépôts. Tout comme le client final dans l'infrastructure DNS n'interroge pas directement les serveurs faisant autorité mais délègue ce travail à un résolveur, un routeur qui s'appuie sur RPKI+ROA interroge un cache validateur qui est chargé de récupèrer, une fois, l'ensemble des objets puis de récupèrer, régulièrement, les nouveaux objets, de les valider avec la cryptographie et de transmettre les assertions contenues dans les ROA aux routeurs.

Ce choix de conception est facile à expliquer : un routeur a pour vocation première de construire une table de routage et de transférer des paquets IP au prochain routeur. Un routeur n'a pas vocation à récupèrer des données et encore moins à les valider avec la cryptographie.

La mise en cache du contenu des ROA se justifie par le fait que les données changent rarement : une autorisation d'annoncer un préfixe ne change pas tous les jours, la vCard d'un Ghostbusters non plus, ... La mise en cache "locale", au préalable, est également importante afin de pouvoir approvisionner un ou plusieurs routeurs en un temps correct et surtout pour limiter l'impact d'un problème qui paralyserait les principales instances de publication.

L'inconvénient de ce choix est que l'on se retrouve avec le problème commun à tout système de mise en cache, à savoir : « à quelle fréquence faut-il rafraîchir les données du cache ? ». Contrairement au

DNS et au TTL des enregistrements, RPKI+ROA ne dispose d'aucun système permettant d'indiquer, à un cache validateur, à partir de quel instant une information devient obsolète et qu'il convient de tenter une nouvelle récupération. Les RFC n'imposent pas une fréquence de rafraîchissement aux caches validateurs. Actuellement, on voit souvent une périodicité de 24 heures. Cela pose des questions vis-à-vis de la sécurité (exemple : combien de temps faut-il avant qu'une révocation devienne effective partout ?).

Qui peut installer un cache validateur ? Les données étant publiques et les logiciels pour le faire étant libres et gratuits, n'importe qui peut installer un cache validateur sur sa machine. Chaque opérateur peut ainsi installer son cache validateur dédié. Pour les vrais déploiements, le RFC conseille néanmoins une structure en strates ressemblant à celle de NTP afin d'éviter de surcharger le système de dépôts. Un exemple d'installation est donné dans le RFC. Le "petit" (c'est relatif) opérateur peut ne pas installer un cache mais utiliser ceux de ses transitaires. Un gros opérateur peut avoir plusieurs caches, se synchronisant entre eux de manière hiérarchique (comme NTP) et utiliser les caches de ses transitaires en cas de problème avec les caches internes.

5.2.5.1 Problème d'amorçage Pour valider les objets, un cache validateur les récupére tous dans les instances de publication. Les certificats racines font quasiment exceptions car ils sont distribués à la fois dans le système de dépôts et hors bande. Typiquement, le certificat racine de chaque RIR est distribué dans un point de publication. Mais récupérer ces certificats comme le reste des objets, c'est-à-dire, dans le système de dépôts, via un canal non sécurisé, n'apporterait aucune sécurité. Il faut donc les récupérer en dehors du système de dépôts.

Un format, « RPKI Trust Anchor Locator », a été normalisé afin de savoir quelles sont les informations à communiquer à un cache validateur pour que celui-ci puisse récupèrer un certificat racine. Ce format est banal : un URI rsync et la clé publique de ce certificat racine encodée en base64.

La distribution d'un TAL se fait sur un autre canal, possiblement sécurisé. Exemples :

— L'ARIN distribue le TAL de son certificat racine par mail ce qui représente un canal alternatif.

— Le RIPE distribue les TAL de tous les RIR, sauf l'ARIN, dans son logiciel qui assure les fonctions de cache validateur. Ce téléchargement se fait en HTTP (canal alternatif) sécurisé par TLS (on fait donc confiance à l'AC x509 classique choisie par le RIPE [23], DigiCert).

Muni du TAL, un cache validateur utilise l'URI indiqué dans celui-ci, récupère le certificat racine et compare la clé publique qui y ai contenu avec celle indiquée dans le TAL. Si la comparaison donne un résultat positif, alors le certificat a une très forte probabilité d'être le véritable certificat non altéré.

23. On fait aussi confiance à toutes les AC x509 présentes dans le magasin de certificats d'AC côté client mais c'est un autre sujet. Voir : `http://www.bortzmeyer.org/6698.html` .

5.2.5.2 Validation des objets et des certificats

Une fois que le cache validateur a récupéré tous les objets de RPKI+ROA (certificats d'AC, objets, ...), il peut commencer le processus de validation. Chaque objet validé est mis en cache. Le processus est détaillé en totalité dans les différents RFC et il est dérivé de celui de PKIX. Je vais simplement résumer les grandes lignes :

- Le cache validateur doit vérifier que tous les points de publication sont complets et intacts à l'aide des Manifest. Il faut bien entendu vérifier préalablement que les Manifest sont valides ...
- Pour chaque objet, il faut construire une chaîne de validation jusqu'à un certificat racine connu : valider la syntaxe des objets et des certificats (sont-ils au format normalisé ?), valider les signatures de chaque certificat, vérifier qu'ils ne sont pas révoqués, faire des contrôles de sémantique (ce certificat a-t-il l'autorisation de déléguer telle ressource à cet autre certificat ?), ...

5.2.6 Des routeurs

Les derniers maillons de la chaîne RPKI+ROA sont les routeurs de la Default-Free Zone. Pour appliquer un traitement en fonction de la validité des annonces, les routeurs doivent avoir connaissance du contenu des ROA valides. Pour rappel, les routeurs communiquent règulièrement avec un ou plusieurs cache validateurs pour récupèrer l'intégralité des autorisations valides au fil de l'eau.

5.2.6.1 Un nouveau protocole

L'échange entre un cache validateur et un routeur se fait selon un nouveau protocole : RPKI To Router (RTR). RTR est un protocole qui permet de distribuer les autorisations valides entre un cache validateur et un routeur. Cela signifie que RTR n'est pas spécifique à RPKI+ROA mais peut être utilisé dans d'autres protocoles de sécurisation du routage : seul le cache validateur change et peut, potentiellement, récupérer les assertions de routage (les ROA actuels) via un autre biais qu'un système de dépôts rsync.

Ce protocole, basé sur TCP, se veut simple (afin de maximiser les probabilités qu'il puisse être implémenté dans tous les routeurs existants) et extensible (pouvoir s'adapter à IPv8 et aux ASN sur 64 bits, en gros). Pour qu'un routeur puisse utiliser RPKI+ROA, il lui suffit d'un client RTR et d'espace mémoire pour stocker les assertions valides (format préfixes <-> ASN). Un routeur peut utiliser plusieurs caches validateurs et un cache validateur peut servir plusieurs routeurs en même temps.

5.2.6.2 Format des échanges RTR

Le format des échanges RTR fait penser à un transfert de zone DNS dans lequel les associations préfixes <-> ASN seraient les données de la zone : un transfert RTR peut être total ou incrémental avec l'utilisation d'un serial marquant la dernière version envoyée au routeur, le routeur est notifié quand de nouvelles données sont disponibles, il peut faire la requête de récupération ou ignorer la notification et récupérer les données selon une périodicité définie.

L'entête commun à la majorité des messages RTR est le suivant :

— Version du protocole sur 8 bits. Actuellement fixée à 0.

— Type du message sur 8 bits : 0 - une notification ; 1 - demande de transfert incrémental ; 2 - demande de transfert intégral ; 3 - confirmation, par le cache validateur, de la bonne réception de la demande du routeur ; 4 - le cache validateur envoie une autorisation concernant un préfixe IPv4 ; 6 - le cache validateur envoie une autorisation concernant un préfixe IPv6 ; 7 - le cache validateur indique qu'il a transmis toutes les informations en sa possession ; 8 - le cache validateur n'est pas en mesure de répondre à une demande de transfert incrémental ; 10 - reporter les erreurs à l'autre partie.

— Session ID sur 16 bits : généré par le cache validateur, il permet au routeur de l'identifier. Cela permet de détecter des problèmes : le cache validateur a été remis à 0. Cela permet aussi à un routeur de reprendre une session interrompue. Ce nombre est fixé à 0 lors d'une demande de transfert total, lors des transferts et lorsque le cache validateur indique ne pas pouvoir répondre à une demande incrémentale.

— Taille sur 32 bits : nombres d'octets de toute la trame (en-tête + contenu).

Le reste des champs des messages est tout aussi simple :

— Une demande de transfert intégral (type 2), la confirmation de bonne réception de la demande (type 3) et la notification d'une impossibilité de transfert incrémental (type 8) ne contiennent aucun champ supplémentaire.

— Une notification (type 0), une demande de transfert incrémental (type 1) et fin du transfert (type 7) contiennent un seul champ supplémentaire : serial. Ce nombre sur 32 bits, identique au DNS, indique simplement la version des données possédées par le cache validateur. Il est incrémenté par le cache validateur uniquement quand de nouvelles données ont passé l'étape de validation.

— Les messages qui permettent l'échange d'associations préfixes <-> ASN contiennent des champs supplémentaires :

  — Drapeaux sur 8 bits : 0 si l'association a été supprimée. 1 pour une nouvelle association.

  — Longueur du préfixe sur 8 bits. Valeurs acceptables : 0 à 32 pour IPv4 et 0 à 128 pour IPv6.

  — Max Length sur 8 bits. Valeurs acceptables : 0 à 32 pour IPv4 et 0 à 128 pour IPv6.

  — Préfixe IP sur 32 ou 128 bits.

  — ASN sur 32 bits.

— La trame permettant de reporter les erreurs est légèrement différente : le numéro de session est remplacé par un code d'erreur normalisé. Il contient aussi des champs permettant de copier à l'autre partie le message qui a causé l'erreur ainsi qu'un message d'erreur en texte clair.

Une communication entre un cache validateur et un routeur est toujours à l'initiative de ce dernier. Au sein d'une communication, l'échange d'informations est aussi à l'initiative du routeur. Le cache validateur ne peut envoyer que des notifications de sa propre initiative. Je vais présenter deux échanges typiques entre un cache validateur et un routeur.

L'initialisation d'une communication commence par la fameuse poignée de main TCP. Le routeur fait ensuite (pas nécessairement directement après l'ouverture TCP) une demande de transfert total. Le cache validateur confirme la réception de la demande et transfert, dans des messages séparés, les 0 ou plus autorisations valides qu'il possède. Le cache validateur envoie ensuite un message « End Of Data » (type 7).

Après l'initialisation, l'échange le plus courant sera le cache validateur qui émet une notification, le routeur qui émet ensuite (pas forcement à la suite de la réception d'une notification) une demande de transfert. Le transfert s'effectuera comme lors de l'initialisation : confirmation de la demande, envoi des autorisations et de l'indicateur de fin des données à transmettre.

Bien entendu, d'autres échanges sont normalisés : indication d'une erreur, reprise en transfert total quand un transfert incrémental est impossible, ...

5.2.6.3 Sécurité de RTR RTR n'inclu aucun mécanisme de sécurité ni n'impose une méthode de sécurisation du canal.

Pourtant, le "dernier kilomètre", la liaison entre un cache validateur et un routeur est, comme tout le reste de la chaîne, susceptible de subir une attaque, d'autant plus si le cache validateur n'est pas local. Ce choix est délibéré de la part des auteurs de RPKI+ROA et se justifie par le fait qu'aucun mécanisme de sécurisation du canal n'est disponible sur tous les routeurs. RPKI+ROA étant la première étape vers un routage sécurisé, si l'on empêche une majorité de personnes souhaitant déployer RPKI+ROA de le faire pour cause d'incompatibilité matérielle, on n'est pas près d'avoir un routage sécurisé.

De plus, les RFC ont séparé routeurs et validateurs pour ne pas surcharger les routeurs avec des calculs cryptographiques. Il serait contre-productif d'imposer ensuite une technique de sécurisation qui demanderait à ces mêmes routeurs de faire des calculs cryptographiques.

Le RFC conseille simplement une suite de techniques de sécurisation du canal : TCP AO, TCP MD5, SSH, TLS, IPsec et la meilleure manière de déployer chacune d'elles tout en indiquant que la préférence des auteurs va à TCP AO dès qu'il sera implémenté sur la majorité des routeurs. En attendant, chacun utilise la méthode qu'il souhaite, même TCP nu, de préférence uniquement sur un réseau local à accès restreint (mais qui suit les recommandations des RFC ?).

5.2.6.4 Que faire des autorisations reçues ? Le routeur tient compte des autorisations reçues (que le RFC nomme liste des « Validated ROA Payload »- liste des VRP) pour construire sa table de routage.

À chaque annonce qu'il reçoit, le routeur cherche, dans une base de données interne, un VRP qui couvre l'annonce. Rappel : il y a une notion de hiérarchie des préfixes IP donc un VRP pour 192.0.2.0/24 couvre une annonce pour 192.0.2.128/26, d'où le terme « couvrir ». S'il trouve une correspondance, il vérifie l'autorisation (l'AS positionné dans l'annonce BGP a-t-il le droit d'annoncer ce préfixe ?).

Cet algorithme peut sortir 3 états :

— Non trouvé : aucun VRP ne couvre le préfixe annoncé.

— Valide : au moins un VRP couvre le préfixe annoncé et l'AS d'un VRP correspond à l'AS de l'annonce.

— Invalide : au moins un VRP couvre le préfixe annoncé mais aucun d'indique l'AS de l'annonce.

L'état de sortie peut ensuite être utilisé comme paramètre pour filtrer les annonces. Toutes sortes de décisions peuvent être appliquées :

— simple remontée d'erreur concernant les origines invalides ;

— affectation d'une préférence locale différente pour chaque état de sortie (exemple : valide > non-trouvé > invalide) ;

— rejet des routes dont l'origine a été marquée comme invalide ;

— ...

Le choix des actions est un choix local qui fait pleinement partie de la politique de routage d'un AS : la validité de l'origine est uniquement un critère supplémentaire pour faire un choix.

5.2.7 Sécurité

Je vais m'attarder sur quelques points de sécurité : algorithmes, révocation, roulement des clés, ...

5.2.7.1 Algorithmes cryptographiques Comme je l'ai déjà dit, RPKI+ROA ne dépend pas d'une paire d'algorithmes cryptographiques (fonction de hachage et algorithme de signature) mais peut en changer suivant les avancées de la cryptanalyse.

Les certificats x509 disposent de champs, « signatureAlgorithm » et « Public Key Algorithm », qui permettent d'indiquer les algorithmes que l'on utilise. Il en va de même pour CMS, à travers les champs « digestAlgorithm » et « signatureAlgorithm ». Rien n'est donc figé.

Le RFC 6485 indique les algorithmes qu'il faut utiliser, SHA256 et RSA avec des clés de 2048 bits, et précise qu'il suffit de le rendre obsolète avec un nouveau RFC pour changer, en théorie, les algorithmes utilisés dans RPKI+ROA. La pratique suivra avec un temps de retard.

Les AC utilisent aussi SHA-1 mais uniquement pour donner, à l'intérieur d'un certificat, un condensat de la clé publique de l'AC signataire et celle du certificat signé.

La procédure pour changer l'algorithme utilisé dans la RPKI est décrite dans le RFC 6916 : les AC commencent à émettre des certificats avec le nouvel algorithme, puis les RP (les validateurs) com-

mencent à pouvoir vérifier le nouvel algorithme, puis les AC arrêtent d'utiliser l'ancien algorithme, puis on peut déclarer l'ancien algorithme mort. Le passage à chaque étape nécessite que tous les acteurs impliqués aient migré, ce qui implique une longue période de coexistence entre les deux algorithmes. Aucune procédure n'est envisagée pour le cas, catastrophique, où de brusques percées de la cryptanalyse obligeraient à un remplacement d'urgence. (source : [BORTZMEYER(2012)])

5.2.7.2 Révocation Une autre problématique de sécurité est la révocation des certificats et des objets.

RPKI+ROA utilise les listes de révocations x509 profilées (usage restreint, comme les certificats). Une telle liste porte sur tous les objets de l'AC, pas juste sur une partie comme c'est possible en x509 PKIX. Il y a donc une seule liste par AC. Elle contient le numéro de série de chaque certificat révoqué associé à la date et l'heure de la révocation. La liste est signée avec la clé privée associée au certificat de l'AC. Elle est mise à disposition dans le point de publication de l'AC : on la reconnait grâce à son suffixe normalisé « crl ».

Notons que les certificats d'AC inclus un champ obligatoire pour indiquer l'URI complet à laquelle récupérer la liste des révocations associée à ce certificat : elle peut donc, en théorie, être hébergée dans un autre point de publication.

Les certificats d'AC sont révoqués par l'AC supérieure. Les objets (ROA, Ghostbusters, ...) sont révoqués en révoquant le certificat EE associé à l'objet. La révocation se fait en ajoutant de nouvelles entrées dans la liste des révocations d'une AC. Pour que la révocation devienne effective (globalement, partout), il faut que la liste (et le Manifest de l'AC) soit récupérée et validée par tous les caches-validateurs puis que les routeurs soient informés de la suppression des assertions.

Tout comme un Manifest, une liste de révocation a un intervalle de validité et il faut donc la re-générer à intervalle régulier, soit pour s'adapter aux nouvelles révocations, soit pour renouveler la signature avant expiration.

5.2.7.3 Roulement des clés Un débat agite les milieux de la cryptographie depuis de nombreuses années (depuis toujours ?) : doit-on changer les paires de clés et selon quelle périodicité ? En la matière, il y a deux écoles de pensée :

— Celle qui dit que les remplacements doivent être systématiques et fréquents, pour qu'ils deviennent la routine ; elle se sépare en deux sous-écoles, une qui favorise les remplacements périodiques et une qui préfère les faire au hasard,

— Celle qui dit que les remplacements ne doivent se faire que s'il y a une bonne raison : une forte suspicion que la clé privée a été copiée, par exemple.

RPKI+ROA n'échappe pas à ce débat et un RFC est consacré à la question du roulement des paires de clés. Ce RFC ne prend pas parti dans le débat mais indique que les roulements de clés

doivent se faire en prenant en considération la charge occasionnée pour le système de dépôts et les caches validateurs et que les roulements doivent, de ce fait, être effectués à un intervalle raisonnable ... qui n'est ni imposé ni même conseillé.

Notons que seule la paire de clés associée à un certificat d'AC peut être changée. Cela se fait de manière automatique pour les objets si l'on suit la recommandation « un certificat EE signe un et un seul objet » et que l'on a donc supprimé la clé privée après signature, comme conseillé.

Un roulement des clés dans RPKI+ROA recoupe les mêmes problématiques que l'on rencontre avec un roulement de clés DNSSEC compte tenu d'une architecture commune (un système de distribution et des caches). Dans les deux systèmes, les RFC indiquent que la priorité doit être accordée au maintien du système afin que les routeurs ne puissent pas prendre de mauvaises décisions causées par ce roulement de clés. Cela signifie qu'il faut toujours qu'il y ait au moins une chaîne de validation intacte. Dans les deux systèmes, la procédure conseillée est celle de la pré-publication : mettre à disposition des caches validateurs le nouveau certificat avant le roulement effectif.

Basiquement, un roulement de clés RPKI+ROA se déroule comme suit, l'algorithme est entièrement détaillé dans le RFC 6489 :

1. L'AC génère une nouvelle paire de clés et une demande de certification.
2. Elle transmet sa demande de certification à l'AC supérieure. Elle attend que son certificat soit publié dans le point de publication de l'AC supérieure.
3. Elle génère une liste de révocation vide ainsi qu'un Manifest et le publie dans son point de publication (qui contient déjà les objets associés à l'ancienne paire de clés, un point de publication ne changeant pas pendant un roulement de clés).
4. L'AC entre dans une période d'attente pour laisser le temps aux caches de récupèrer le nouveau certificat et les premiers objets (la liste des révocations et le Manifest). Cette étape doit être contournée s'il s'agit d'un roulement des clés en urgence c'est-à-dire si la paire de clés précédente a été compromise. Le RFC conseille un temps d'attente au moins égal à 24 heures. Ce qui est peut-être insuffisant : aucune périodicité n'ayant été définie pour imposer à un cache validateur de rafraîchir son cache. De plus, 24 heures peuvent être insuffisantes pour les caches "en strate" : cache1 peut prendre 24 heures pour rafraîchir son cache mais cache2 peut aussi prendre 24 heures pour rafraichir son cache en se synchronisant sur cache1 ... Durant cet intervalle, l'AC doit resigner tous les objets (non expirés, non révoquée, bien sûr) qu'elle a signés avec son ancienne paire de clés. Mais elle ne doit pas les publier.
5. L'AC publie tous les objets signés avec sa nouvelle paire de clés. Il faut évidemment re-générer les deux Manifest (celui correspondant à la nouvelle paire de clés et l'autre). De l'ancienne instance, il ne doit rester qu'un Manifest et la liste des révocations.

6. Générer une demande de révocation de l'ancien certificat d'AC et la transmettre à l'AC supérieure.
7. Quand la révocation est effective (liste des révocations de l'AC supérieure mise à jour), l'AC doit supprimer son ancien Manifest et son ancienne liste de révocation de son point de publication.

En lisant cette procédure, on peut se poser une question : « lorsqu'une AC effectue un roulement de clé, il faut que toutes les AC "filles" resignent leurs objets ? ». Non, un roulement de clés se limite à resigner uniquement les objets de l'AC qui change ses clés. Rappel : le schéma de nommage impose, pour les certificats d'AC un nom de fichier dérivé de la clé publique de l'AC, pas de l'AC "parente". Donc une AC qui fait un roulement de sa paire de clés resignera le certificat d'une AC "fille" mais le nom de ce certificat ne changera pas. Donc la hiérarchie n'est pas cassée : les champs « AIA » des objets de l'AC "fille" pointeront toujours sur le bon certificat.

5.2.7.4 Attaque par rejeu ? La non-possession des clés privées d'une AC donnée ne permet pas de forger des objets avec une signature valide. En revanche, rien n'interdit à un potentiel attaquant de rejouer un objet durant son intervalle de validité. En effet, la signature ayant été générée par la partie privée du certificat CA ou EE et l'intervalle de validité du certificat étant en cours, la signature sera valide donc le rejeu sera accepté par le cache validateur.

RPKI+ROA a des mécanismes pour limiter ce type d'attaque : un mécanisme de révocation et les Manifest. Rejouer un objet (au sens large : ROA, Ghostbusters, certificat, ...) invalide mais non expiré et non révoqué reste possible presque sans limites mais cette situation a une probabilité d'existence plutôt faible. Les Manifest permettent de contrer les tentatives basiques puisqu'ils contiennent un condensat cryptographique de chaque fichier présent dans un point de publication. Le RFC qui leur est consacré indique que les Manifest ne permettent pas nécessairement de détecter les attaques par rejeu précises qui se déroulent dans un intervalle de temps très fin.

À mon avis, deux facteurs favorisent les attaques par rejeu dans RPKI+ROA.

D'une part, le traitement des Manifest n'est pas assez exigeant. En effet, à chaque situation douteuse (pas de Manifest dans le dépôt, Manifest expiré, problème de correspondance entre la liste de fichiers contenue dans le Manifest et la liste des fichiers réellement présents dans un point de publication, non-correspondance entre les condensats cryptographiques, ...), le RFC demande une simple remontée d'information. Cette attitude prudente favorise la récupération d'objets douteux. Mais être plus exigeant signifie aussi être moins tolérant aux erreurs humaines, ce qui peut amener les routeurs à faire de mauvais choix, ce que veulent éviter les auteurs de RPKI+ROA.

D'autre part, l'absence de contraintes sur le temps de diffusion des objets simplifie les attaques par rejeu. En effet, il n'existe pas de mécanisme comparable au TTL du DNS permettant d'indiquer aux caches validateurs quand rafraîchir les données qu'ils possèdent et les RFC n'imposent aucun délai pour rafraîchir les caches, ni celui d'un cache validateur ni celui d'un routeur.

Puisque la révocation effective d'un objet commence réellement quand les autorisations concernées sont supprimées de tous les routeurs qui participent à RPKI+ROA, il faut considérer le temps de diffusion de la liste des révocations (crl) et du Manifest (sans quoi un simple rejeu additionnel d'une version antérieure de la liste de révocations permet de mener à bien l'attaque). Il faut donc prendre en compte :

— le temps de génération de ces deux objets ;
— le délai de mise à disposition de ces deux objets dans un point de publication. Ce délai peut être variable, par exemple : une AC "mère" qui doit traiter la demande de révocation d'une AC "fille".
— le délai avant que tous les caches validateurs récupèrent les objets et les valident ; certains RFC parlent de 24 heures mais ce n'est même pas une recommandation. Et s'il y a plusieurs caches en cascade ?
— la périodicité avec laquelle le routeur demande les changements au cache validateur si jamais ce routeur ne prend pas en compte les notifications. Là encore les RFC ne fixent pas ce délai.

Durant cet intervalle de temps, qui peut donc être de plusieurs heures, une attaque par rejeu est possible même si elle devient partielle dès que les premiers caches récupèrent la révocation et le Manifest.

Néanmoins, une attaque par rejeu présente de fortes limites.

D'une part, les points d'attaque sont limités. L'attaquant doit, soit corrompre une instance de publication, ce qui, au vu des acteurs en place (RIR), n'est pas gagné d'avance, soit l'attaquant réalise un man in the middle entre le système de dépôts et un cache validateur et là, on est dans une attaque spécifique et ciblée, pas dans une attaque massive. Il y a aussi la furtivité de l'attaque à prendre en compte.

D'autre part, il faut contrôler la ressource détournée : rejouer un ROA qui dit « l'AS 64496 peut être à l'origine du préfixe 198.18.0.0/24 » a un intérêt limité. Si l'attaquant souhaite isoler un réseau en faisant rejeter sa route par les autres routeurs, il faut encore que les routeurs qui valident avec RPKI+ROA suppriment les routes invalides, ce qui n'est pas évident : il faut qu'un nombre suffisant de caches validateurs récupèrent l'objet rejoué, le valide, le "propage" aux routeurs qui auront certainement une politique plus coulante qu'un brutal rejet des routes. Pour récupérer le trafic, l'attaquent doit être l'AS 64496, ce qui est tout aussi compliqué : un ASN ne change vraiment pas souvent de titulaire. De plus, les préfixes ne changent pas tous les jours d'AS d'origine ce qui limite fortement les possibilités de rejeu.

Enfin, les remarques de la partie « Des attaques concrètes » restent valables : de meilleures techniques (plus efficaces en nombre d'utilisateurs impactés et/ou plus accessibles) ou des techniques aussi

complexes mais plus efficaces ne sont-elles pas à disposition ? Il me semble que si : DDoS des instances de publication, attaque BGP modifiant le chemin d'AS mais pas l'origine, ...

5.2.8 Logiciels

Comme RPKI+ROA est multifacettes, plusieurs types d'outils sont nécessaires.

D'abord, pour ceux qui distribuent les ressources, il faut générer les certificats ainsi que les objets cryptographiques. On retrouve les besoins des autorités de certificats x509 "classiques" couplés à des besoins de gestion des adresses IP (IPAM) : il faut générer les certificats et les objets, les renouveler, éventuellement les révoquer, vérifier l'association entre un préfixe, son certificat et ses objets, ...

À l'heure actuelle, les deux logiciels libres d'IPAM que je connais, Netdot et Netmagis ne supportent pas la création de certificats et d'objets. Il en va de même pour les nombreux IPAM que j'ai trouvés durant mes recherches : Infoblox, Cisco Network Registrar, BT Diamond IP, NIPAP, ...

En ce qui concerne les logiciels spécialisés, on trouve Local Certification Service du RIPE mais il s'agit d'un proof-of-concept qui ne supporte pas, par exemple, la création de ROA. En solution complète, on trouve les CA tools du projet rpki.net. Pour les plus déterminés, l'utilisation d'OpenSSL (compilé avec les extensions du RFC 3779) reste possible mais on n'est plus dans un contexte de production.

Ensuite, il faut distribuer les objets. Soit on fait publier ses objets dans une instance de publication existante, soit on héberge son instance. Le seul protocole obligatoire est rsync. Il suffit de ce logiciel libre pour faire un point de publication.

La solution la plus simple à l'heure actuelle consiste à se reposer sur les services "cloud" des RIR, par exemple : Resource Certification Service du RIPE.

Pour les caches validateurs, au moins trois implémentations sont disponibles : RPKI Validator du RIPE, rcynic du projet rpki.net et RPSTIR de BBN.

Pour les routeurs, un client RTR est inclus, chez Cisco, à partir d'IOS 12.2S, XE 3.5S et XR 4.2.1. Pour Juniper, la même chose est disponible à partir de la version 12.2 de JunOS. Je n'ai rien trouvé concernant l'ajout d'un client RTR chez les autres équipementiers auxquels j'ai pensé (Huawei, Alcatel).

Du côté des solutions en logiciel libre, Quagga permet l'utilisation des ROA et dispose d'un client RTR, via le patch BGP-SRx, depuis sa version 0,99. BIRD dispose aussi de la gestion des VRP depuis sa version 1.3.7 mais de dispose pas d'un client RTR pour mettre à jour automatiquement sa table. OpenBGPd et XORP n'ont aucun support des ROA.

Enfin, pour déboguer ou découvrir, les outils traditionnels sont disponibles :

— looking glass. Exemple : `http://www.labs.lacnic.net/rpkitools/looking_glass/` ;

— des services indiquent, via whois, l'état de l'origine des routes. Exemples : « whois -h whois.bgpmon.net 8.8.8.0/24 » ou « whois -h whois.rpki.net 5.104.72.0/23 » ;
— des routeurs Cisco et Juniper sont librement accessibles pour consulter l'état des routes et faire quelques manipulations de base. Exemple : les accès mis en place par le RIPE : `https://www.ripe.net/lir-services/resource-management/certification/tools-and-resources`.

5.3 Limites

5.3.1 RPKI+ROA ne sécurise que l'origine

RPKI+ROA ne s'occupe, en effet, que de permettre la validation de l'origine d'un préfixe donc, oui, les attaques visant la modification du chemin d'AS restent réalisables même dans un monde où RPKI+ROA est totalement déployé. RPKI+ROA permet, tout de même, de contrecarrer toutes les erreurs humaines, de plus en plus nombreuses chaque année, qui surviennent dans le routage.

La validation du chemin d'AS est bien plus complexe que la sécurisation de l'origine : explosion combinatoire du nombre de possibilités, relations entre opérateurs qui ne sont pas dans les bases des RIR, changements fréquents (des routes apparaissent et disparaissent au rythme des problèmes qui sont la norme sur Internet), performance (chaque routeur doit signer chaque annonce qu'il relaye), ...

RPKI+ROA n'est qu'une étape, la première marche vers un routage inter-AS sécurisé et a toujours été présentée ainsi dans les RFC. C'est justement son point fort : un routage sécurisé est complexe à mettre en place, toutes les solutions qui le propose (S-BGP, soBGP) le prouvent donc on découpe l'impossible en petites étapes franchissables progressivement : on met d'abord en place une infrastructure à clés publiques, on s'en sert pour valider l'origine d'une route et l'on sécurisera le chemin dans une prochaine itération.

La seule ombre au tableau est que, puisque RPKI+ROA n'est qu'une première étape, pourquoi la déployer ? Une attitude passive jusqu'à la conception intégrale de la solution complète risque d'être adoptée par les opérateurs. Ils ont déjà du mal à intégrer IPv6 à leurs réseaux alors RPKI+ROA ...

5.3.2 RPKI+ROA et la pratique

À mon avis, les RFC décrivant RPKI+ROA ne sont pas suffisamment complets d'un point de vue opérationnel. Les délais (délai de diffusion d'un objet, délai de révocation, ...) ne sont pas précisés.

Certains délais (délai avant mise à disposition dans le système de dépôts, délai de prise en compte d'une demande) sont spécifiques aux AC et seront précisés dans le Certification Practice Statement de l'AC, ce document dans lequel chaque AC indique ses procédures, ses délais de traitement, bref, ses pratiques concernant la gestion de ses certificats. Pour les autres délais, c'est le flou.

Ce flou peut faciliter des attaques et ne permet pas de fournir des réponses opérationnelles (combien de temps avant que je puisse annoncer mon sous-préfixe sans lever d'erreur (donc : en combien de temps

le ROA associé sera validé partout) ? Combien de temps avant qu'une révocation devienne effective ? ...)

De plus, selon moi, des mécanismes et des procédures supplémentaires devront être normalisés :

— Les certificats racines seront amenés à être changés (certains d'entre eux ont même déjà changé). Normalise-t-on un mécanisme de suivi et d'intégration de ces certificats, comme dans DNSSEC, ou on se contente d'annoncer les roulements sur les listes de diffusion orientées réseau en espérant que tous les opérateurs lisent et agissent ?

— Les procédures de validation des Manifest sont, pour certaines, trop laxistes et devront être précisées pour pouvoir assurer réellement la sécurité du système de dépôts en toutes circonstances et notamment contre les attaques par rejeu. Mais cela ne peut se faire que difficilement tant que les procédures de création/publication d'objets ne sont pas "maitrisées".

— L'avitaillement de certificats entre deux AC fait l'objet d'un protocole normalisé mais les mécanismes de sécurité attenants sont laissés à la discrétion des AC, ce qui peut poser quelques problèmes d'entente.

Nous sommes au début du déploiement ce qui signifie que les premiers grands déploiements de terrains permettront la capitalisation de retours d'expérience concrets qui donneront lieu à l'écriture de nouveaux RFC orientés sur les pratiques opérationnelles.

5.3.3 RPKI+ROA va exclure des réseaux des Internets

Cette crainte vient d'une comparaison : si, à l'heure actuelle, les opérateurs filtraient les routes en se basant sur les IRR (Internet Routing Registry), il y aurait de nombreux problèmes étant donné que les IRR ne sont pas tenus à jour par ces mêmes opérateurs.

Bien sûr qu'il y a et qu'il y aura des erreurs et de la négligence avec RPKI+ROA puisque ce dernier n'est pas un patch pour l'humain. Actuellement, selon les chiffres de LACNIC, environ 12% des routes dont l'origine est vérifiée par un ROA sont invalides [24]. Notons que ce chiffre s'élevait à environ 60% en décembre 2011 [25] et à environ 40% en février 2012 [26], à la sortie des RFC. À mon avis, cette baisse s'explique principalement par le fait que de nombreuses personnes ont expérimenté "juste pour voir", ce qui a entrainé la création de ROA invalides, avant de supprimer ou modifier leurs essais.

Concernant la comparaison entre IRR et RPKI+ROA, il appartient pleinement aux opérateurs de tenir la RPKI plus à jour que les IRR. Bien sûr, ce n'est pas si simple et certains y verront des coûts récurrents supplémentaires comme avec DNSSEC : il faudra mettre en place, maintenir (révoquer, adapter les ROA à la nouvelle topologie, ...) et réparer.

24. Source : `http://www.labs.lacnic.net/rpkitools/looking_glass/`.
25. `https://labs.ripe.net/Members/waehlisch/one-day-in-the-life-of-rpki`.
26. `https://labs.ripe.net/Members/AlexBand/resource-certification-rpki-in-the-real-world`.

Concernant le fond du problème, la disparition entière de réseaux de l'Internet suite à des erreurs dans les ROA me semble fortement improbable : entre connectivité et sécurité, les opérateurs choisiront la connectivité c'est-à-dire que peu d'entre eux effectueront un filtrage brutal des routes ayant une origine invalide : baisser la préférence locale de la route sera préféré et les réseaux concernés seront donc toujours accessibles. Et pour les opérateurs qui feront le choix d'un filtrage strict des routes invalides, la communication et la réactivité humaine permettront de corriger ces aléas.

5.3.4 Comment apporter la sécurité aux préfixes legacy ?

Comme nous l'avons vu, les préfixes legacy, qui représentent environ 40% de l'espace adressable [27], ont été partagés, au début des années 2000, entre les RIR, en fonction de la localisation géographique des titulaires de ces préfixes. Mais, ce n'est pas pour autant que les titulaires sont devenus membres du RIR qui leur a été affecté, tant s'en faut. Or, les RIR distribuent des certificats de ressources uniquement à leurs membres, c'est-à-dire ceux qui participent aux coûts du RIR via la cotisation annuelle. Dans cette situation, comment sécuriser le routage des préfixes legacy ?

D'une part, il est improbable que les RIR distribuent des certificats de ressources aux titulaires de préfixes legacy non membres. D'autre part, il est improbable qu'une majorité de titulaires accepte de s'enrôler dans un RIR alors que le statut ERX leur offre un certain nombre d'avantages car ils sont en dehors de tout système : pas de frais, indépendance, statut juridique flou, ...

Pour les titulaires de préfixes legacy qui le souhaitent, il leur reste la possibilité d'établir leur propre autorité de certification. Le problème sera de diffuser leur certificat racine : leur autorité sera-t-elle jugée assez légitime par la communauté entière pour être acceptée dans une majorité de caches validateurs ?

Cette problématique n'a pas lieu d'être en IPv6 puisqu'il n'y a pas de préfixes v6 legacy. Néanmoins, comme la cohabitation v4/v6 est amenée à se prolonger, souhaiter un passage à IPv6 pour résoudre ce problème n'est pas une approche suffisante.

5.3.5 RPKI+ROA est complexe

Comme toute technique de sécurisation, RPKI+ROA présente une complexité. De nouveaux rôles sont distribués aux acteurs, de nouvelles procédures sont à prévoir et de nouveaux outils sont à découvrir (la cryptographie et x509 ne sont pas le cœur de métier des opérateurs).

Néanmoins, la complexité est restreinte à un petit nombre d'acteurs : contrairement à d'autres solutions de sécurité comme PGP, TLS ou DNSSEC [28], seuls les opérateurs réseau doivent s'impliquer dans RPKI+ROA, pas les utilisateurs finaux (M. et Mme Michu).

27. Sans tenir compte des préfixes réservés et sans faire de distinction entre les titulaires qui ont accepté de devenir membres d'un RIR et ceux qui refusent.

28. DNSSEC fait actuellement reposer la sécurité "du dernier kilomètre", du résolveur jusqu'à la machine cliente, sur l'utilisateur final. TLS impose une connaissance des grands principes de PKIX et de ses limites pour faire un choix plus éclairer que « oui j'accepte ce certificat car mon navigateur me le demande ». Pas de commentaire pour PGP.

De plus, les aspects les plus déroutants de RPKI+ROA (la cryptographie et les formats utilisés) sont gérés par les logiciels chargés de la création des certificats et des objets et ils sont donc des poids supplémentaires pour les développeurs, pas pour les opérateurs.

Mais bon, des opérateurs qui répètent à outrance que le réseau est une affaire de professionnels (notamment vis-à-vis des FAI associatifs) se doivent de s'adapter et de défier une éventuelle complexité, non ?

5.3.6 Quelle infrastructure pour RPKI+ROA ?

Le fait de savoir quelle puissance est requise par RPKI+ROA et si l'infrastructure suivra et suffira reste une vraie question.

La première question est le temps de diffusion des objets depuis le système de dépôts. Verisign met à disposition une étude [Osterweil et al.(2012)Osterweil, Manderson, White, and McPherson] qui se propose d'évaluer le nombre d'objets qui seront présents dans la RPKI dans le cas d'un déploiement intégral ainsi que le temps nécessaire à la récupération de ces objets. Les estimations sont faites à l'arrondi inférieur. Leurs estimations s'élèvent à 608801 objets et un temps de récupération d'environ 4 jours. Évidemment, il s'agit du délai maximum qui mesure le temps de récupération de l'ensemble des objets, ce qui n'arrive qu'à 2 occasions : mise en place d'un nouveau cache validateur ou crash intégral du cache validateur existant. Le reste du temps, le cache validateur n'effectuera qu'un téléchargement partiel portant uniquement sur les nouveaux objets.

Néanmoins, certains points de cette étude m'intriguent :

— L'étude part du principe que chaque AC doit avoir son objet Ghostbusters. Dans la théorie, pour faciliter le débogage, ce postulat serait vrai. Dans la pratique, à mon avis, on sera loin de l'association 1 AC = 1 objet Ghostbusters.

— L'étude estime 23 routeurs BGP par AS ce qui ne me semble pas refléter la réalité de la majorité des AS : on parle de 2 à 10 routeurs de bordure par AS pour la plupart des AS et plusieurs milliers pour les tiers 1. Ce chiffre n'a aucun impact sur les calculs concernant RPKI+ROA seul mais c'est intéressant à souligner.

— L'étude indique que chaque point de publication peut contenir ou pas un Manifest selon la structure de l'instance, ce qui est assez inexact. Selon le RFC, 1 CA = 1 Manifest. De plus, même si l'on garde un système de dépôts constitué uniquement de 5 instances de publication, celles des RIR, il y a, de toute façon, dans la structure actuelle, 1 Manifest par AC et un dossier (point de publication) par AC. Une structure totalement "à plat" n'est pas envisagée à l'heure actuelle et serait chaotique. Donc, si chaque AS hébergeait lui-même son instance de publication, cela ne rajouterait pas 42 000 Manifest comme l'indique l'étude.

— Le temps de récupération me semble exagéré. Il y a actuellement environ 17000 objets dans le système de dépôts. Pour 608800 objets, l'étude prévoit 382300 secondes. Pour 17000 objets, on devrait donc avoir environ 10700 secondes soit presque 3 heures. Pour rendre l'estimation plus conforme aux calculs de l'étude, il faudrait compter le temps de parcours des dossiers. Même avec une connexion ADSL limitée à 250 ko/s en réception, je suis loin d'avoir mis 3 heures pour récupérer ma copie initiale du système de dépôts.

Une fois que les caches validateurs ont récupéré tous les objets, ils doivent les valider. Cette phase consomme actuellement environ 2 minutes sur un PC portable classique donc ça reste accessible à n'importe quel serveur accessible aux opérateurs. Je pense que la puissance nécessaire pour traiter tous les objets de la RPKI, dans le cas d'un déploiement total, sera accessible à tout opérateur réseau car la capacité de traitement de tout équipement de cesse d'augmenter au fil du temps.

La consommation mémoire du cache validateur et celle de la table des autorisations d'un routeur m'intéressent beaucoup plus.

D'après mes observations, RPKI Validator consomme environ 500 Mio pour environ 3000 ROA.

Pour se faire une idée de la consommation mémoire de la table des autorisations sur un routeur, on peut regarder le routeur Cisco mis à disposition par le RIPE et constater qu'environ 3000 préfixes IPv4 récupérés depuis un même cache validateur consomment environ 320 Kio. Donc l'intégralité des préfixes IPv4 actuellement annoncés occuperait environ 50 Mio supplémentaires auxquels il faut ajouter l'espace nécessaire pour IPv6. Je n'ai pas assez de recul pour estimer si 50 Mio est une quantité de mémoire conséquente pour un routeur de bordure de réseau. Je constate simplement que l'accroisement constant de la taille de la table de routage globale inquiète et amène à réflechir à des solutions (exemple : Locator/Identifier Separation Protocol).

Je suis tout aussi incapable d'estimer le surplus de temps qu'engendre RPKI+ROA lors de la construction d'une table de routage sur un routeur de cœur de réseau. J'imagine néanmoins que les développeurs utiliseront des structures de données adaptées.

5.3.7 RPKI+ROA donne un pouvoir nouveau aux RIR

Ce point est certainement celui qui a fait grincer le plus de dents.

Sur ce point, une comparaison entre DNSSEC et RPKI+ROA n'est pas recevable. DNSSEC confirme le pouvoir de ceux qui le concentraient déjà : oui le gouvernement des États-Unis a toujours eu et voulu garder la main sur la racine du DNS ; oui, l'ICANN concentre un pouvoir ; oui, Verisign, de par sa forte présence dans le DNS (opérateur de la racine, des TLD « net. » et « com. », et des rôles dans d'autres TLD dont « edu. » et « gov. »), concentre du pouvoir. Il conviendrait de nuancer ces pouvoirs mais ce n'est pas le sujet de ce rapport. RPKI+ROA, quant à lui, donne un nouveau pouvoir opérationnel aux RIR.

Sans RPKI+ROA, les RIR n'ont aucun pouvoir pour priver une organisation d'une de ses allocations : un RIR peut toujours effacer les données de ses bases de données (whois, IRR, ...), cela n'a aucune conséquence pratique. En effet, le titulaire "radié" peut continuer à annoncer le préfixe sans contraintes (sauf si les opérateurs se mettent à le filtrer, ...). Le RIR peut re-allouer le préfixe mais cela nuirait plus au nouveau titulaire qu'à l'ancien et le RIR ne prendra pas le risque.

Avec la RPKI, un RIR peut révoquer le certificat de ressource d'une allocation. Cela aura pour conséquence que les caches validateurs ne pourront plus trouver une chaîne de validation et rejetteront donc le ROA. Le préfixe passera donc d'un état « valide » à un état « non trouvé » (et non pas « invalide » !). De plus, actuellement, les RIR s'occupent aussi des instances de publication principales de RPKI, ce qui leur confère un pouvoir supplémentaire.

Les RIR étant des organisations localisées géographiquement, ils peuvent subir la pression des gouvernements, de la justice voire de tout autres groupes de pression afin qu'un RIR invalide une allocation. RPKI+ROA donne bien un nouveau pouvoir opérationnel aux RIR mais il convient d'en nuancer la portée.

D'abord, il faut examiner les précédents pour se faire une opinion. La seule affaire connue est DNSChanger. En novembre 2011, les autorités enquêtent sur le malware DNSChanger. Le suspect, Russian Business Network, est titulaire de préfixes IP dont certains ont été alloués par l'ARIN et d'autres par le RIPE. Les autorités souhaitent que les enregistrements des allocations ne puissent être ni transférés ni modifiés. Une cour pénale états-unienne demande à l'ARIN de prendre les mesures nécessaires. Le FBI intervient auprès de la police néerlandaise afin qu'elle exige la même chose du RIPE (le RIPE étant une association de droit néerlandais). L'ARIN a obtempéré sans aucune remise en question. Le RIPE a obtempéré le temps d'obtenir un avis sur cette question. Apprenant qu'un ordre de la police néerlandaise ne suffit pas, que la demande d'une juridiction étrangère doit être soumise à un tribunal néerlandais avant application éventuelle et que la demande ne s'appuyait sur aucun fondement légal sérieux, le RIPE a suspendu, en janvier 2012, son action prise à la demande des autorités.

Il y a eu un précédent sans RPKI+ROA et les réactions de deux RIR ont été très différentes. Il me semble que RPKI+ROA, une simple technique, n'aurait pas changé les décisions humaines prises par ces deux RIR.

Ensuite, il faut comprendre que la marge de manœuvre des RIR n'est pas énorme : l'action sera remarquée et contrecarrée si c'est jugé nécessaire par la communauté Internet. Ces humains réagiront à l'agression comme ils l'ont toujours fait : on se rappellera des erreurs BGP corrigées en quelques heures, des "effets flamby" et des opérations de mass-mirroring. Certaines personnes imaginent des scénarios impossibles dans lesquels le FBI demanderait, à un RIR, et pas forcement à celui responsable du

préfixe jugé "malveillant", de générer un faux ROA associant, probablement, le préfixe à l'AS 0, afin d'interdire l'annonce du préfixe. Ce type d'action est impossible car cela remettrait en question la légitimité en tant qu'AC du RIR qui aura collaboré avec les autorités et que son certificat racine pourrait être ignoré par les opérateurs (même si, il faut bien se l'avouer, très peu réagiront de la sorte).

De plus, il faut bien comprendre que ce n'est pas parce qu'un ou plusieurs RIR prennent une décision (révoquer un certificat, émettre un faux ROA, ...) qu'elle sera suivie d'effet dans la pratique. En effet, il faut se souvenir que les ROA ne sont qu'un paramètre parmi d'autres permettant de prendre une décision de routage locale. Le choix d'accepter ou de refuser l'annonce d'un préfixe "suspendu" par un RIR revient donc aux opérateurs, pas au RIR ! J'ai déjà exprimé ma pensée sur le fait que les opérateurs privilégieront connectivité à sécurité et ne filtreront donc pas les routes dont l'origine est marquée comme « invalide ». Ils le feront encore moins avec les routes dont l'origine est marquée comme « non trouvé ». Même en admettant qu'un opérateur filtre brutalement les routes, il est composé d'humains qui réagiront s'ils jugent abusive la décision de bloquer un préfixe [29]. La décision des RIR ne s'appliquera donc pas nécessairement sur le terrain.

Enfin, d'autres personnes expliquent que le RIPE a réussi à trouver du matériel légal pour repousser la demande de la police mais qu'il ne faut pas trop compter dessus : les lois peuvent changer.

On arrive à une problématique qui n'est plus technique et dont la solution est simple : chaque personne vivant en démocratie a la démocratie qu'elle se donne les moyens d'avoir. En d'autres termes : « Méfie-toi de la dictature qui sommeille. Le bruit des bottes est un mauvais réveil ». La technique ne peut apporter de solutions à un problème politique (et inversement d'ailleurs).

Encore d'autres personnes expliquent que RPKI+ROA met en danger les aspirants à la liberté dans les dictatures. Pourtant les dictateurs utilisent actuellement des moyens plus efficaces que de faire pression sur un RIR qui n'est même pas situé sur leur zone d'influence pour espérer obtenir la révocation d'un certificat. Exemples : le bricolage BGP et DNS se pratique régulièrement en Chine sans RPKI+ROA. Hosni Moubarak avait éteint temporairement Internet en Égypte en profitant de relations avec les principaux FAI locaux. La même chose s'est produite en Syrie en profitant d'un monopole local dans le secteur des télécommunications. De plus, des solutions techniques permettront toujours de venir en aide aux peuples opprimés et RPKI+ROA et même BGPSec n'y changeront rien.

En parlant de démocratie, RPKI+ROA est une norme produite à l'IETF qui est, pour rappel, un organisme de normalisation ouvert à toute personne souhaitant s'investir. De plus, au RIPE, le choix de continuer le déploiement de RPKI a été soumis au vote des membres lors du RIPE General Meeting de novembre 2011 [30].

29. Bien sûr, tous les opérateurs réseau sont loin d'être éthiques mais ce n'est pas le cas de tous. À chacun de choisir son opérateur.

30. Néanmoins, je reconnais qu'il faut des moyens et du temps pour participer à toutes les rencontres d'un RIR.

6 Mise en œuvre

Pour illustrer les propos de ce rapport, j'ai décidé de réaliser une maquette portant sur BGP et RPKI+ROA. Elle est récupérable à cette adresse : `http://www.guiguishow.info/wp-content/uploads/2013/09/RPKI-ROA/Maquette/Maquette-RPKI-ROA.tar`. L'archive contient la maquette au format compressé LZMA et les instructions pour la décompresser et l'utiliser. Pour reproduire tout ou partie de la maquette à partir de zéro, voir le tutoriel en annexe 1.

6.1 Objectifs

D'une part, une maquette permet d'illustrer :

— quelques-uns des risques auxquels nous sommes confrontés avec BGP ;

— la solution normalisée à l'IETF qui apporte une réponse à ces problèmes : RPKI+ROA ;

— les limites de cette solution.

D'autre part, l'approche pratique permet d'apprendre à mettre en œuvre les nouveaux composants logiciels relatifs à RPKI+ROA et les nouvelles procédures associées. La mise en pratique permet aussi de s'assurer que l'on a bien compris la théorie sous-jacente (on doit être capable de répondre à des questions comme « ce comportement observé est-il attendu ? »).

De plus, une maquette permet aussi de montrer que RPKI+ROA est déployable, sous conditions, dès aujourd'hui car des solutions logicielles existent et sont fonctionnelles.

De manière plus précise, ma maquette a pour vocation d'illustrer :

— les problèmes les plus souvent rencontrés avec BGP à savoir les détournements de préfixes et de sous-préfixes même si les cas connus sont des incidents et non des attaques. Les types angles d'attaques (modification du chemin d'AS, Kapela et Pilosov, ...), du fait de l'absence de cas concrets, me semblent être moins prioritaires pour être illustrés.

— le fonctionnement de RPKI+ROA, les interactions entre ses différents composants ainsi que l'apport de sécurité que cette solution procure.

— les différentes politiques locales que l'on peut appliquer en fonction du résultat de la validation RPKI+ROA et leurs conséquences sur le gain de sécurité. J'ai choisi de mettre en pratique uniquement les politiques qui seront utilisés par les opérateurs : « prefer-valid » (les annonces valides sont préférées aux annonces inconnues qui sont elles-mêmes préférées aux annonces invalides) et « prefer-valid » + « ignore-invalid » (les annonces valides sont préférées aux annonces inconnues).

6.2 Présentation technique

6.2.1 Vue globale

On cherche à obtenir une architecture réaliste (comme si l'on regardait une petite partie d'Internet avec une loupe) qui comprend :

— une AC donc une instance de publication ;
— plusieurs caches validateurs ;
— des AS qui créent des objets cryptographiques et d'autres qui ne le font pas ;
— des AS qui font de la validation et d'autres qui n'en font pas ;
— une ou plusieurs fausses annonces de type détournement de (sous-)préfixe.

Il faut alors se demander quelles sont les topologies les plus courantes et/ou les plus intéressantes d'un point de vue de la RPKI :

— PI multihomée : cas courant et intéressant. L'AS 64500 représente cette situation.
— PI "single" : configuration plutôt rare. Au niveau de la RPKI, cela est équivalent à un préfixe PI multihomé : l'acteur disposera d'un certificat et créera ses objets cryptographiques. Cette situation n'est pas représentée sur ma maquette.
— PA multihomée : cas le plus intéressant. Soit l'acteur a un ASN, ce qui est conseillé, soit non. S'il n'a pas d'ASN, le LIR peut générer tous les objets cryptographiques.
  S'il a un numéro d'AS, soit le LIR peut lui distribuer un certificat soit lui créer les objets cryptographiques.
  Dans ma maquette, je prendrai le cas d'un acteur ayant un préfixe PA et un ASN dont le LIR distribue un certificat car cela crée un niveau supplémentaire de hiérarchie dans la RPKI. L'AS 65540 de la maquette représente cette situation.
— PA single : cas intéressant à mettre sur la maquette pour rappeler qu'un tel acteur ne fait pas partie de la RPKI : il n'a aucun objet à créer, tout est pris en charge par son LIR. L'acteur "Opas" de la maquette représente cette situation.

Le schéma suivant représente l'architecture que j'ai mise en œuvre :

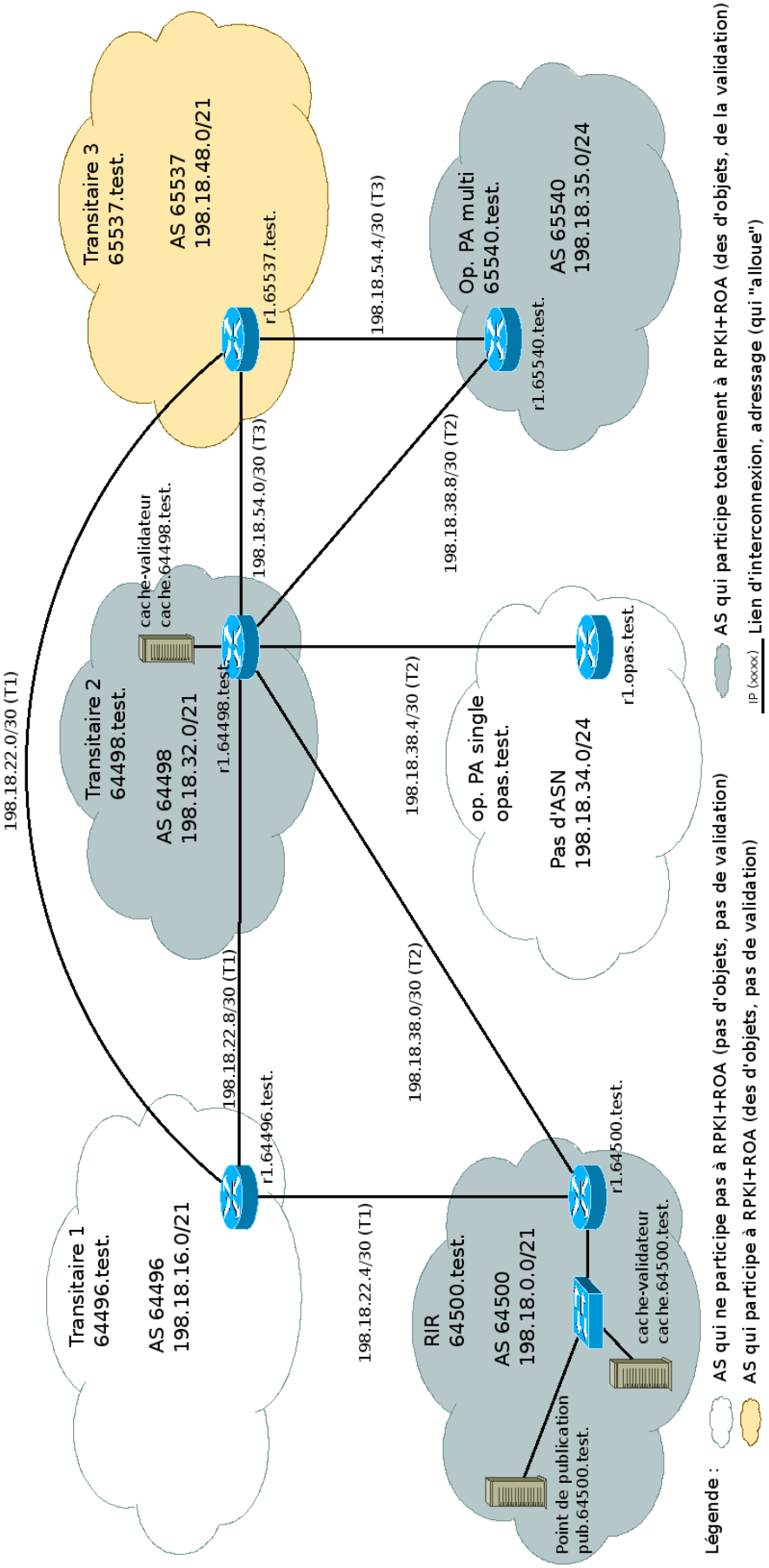

FIGURE 5 – Schéma de ma maquette.

Cette maquette permet de produire les 3 états de validation possibles : inconnu (T1), invalide (T3, opPaMulti), valide (le reste). Elle permet aussi de tester les différentes politiques de validation réalistes : pas de validation (T1, T3), « prefer-valid » (RIR, T2), « prefer-valid »+« ignore-invalid » (OpPaMulti).

6.2.2 Adressage et nommage

En ce qui concerne les numéros d'AS, j'ai décidé de les choisir dans la plage d'ASN réservés à l'IANA pour la documentation : 64496-64511,65536-65551.

On considère que le RIR de la maquette a reçu les allocations suivantes : 64496-64500 et 65536-65540. Soit 10 ASN. Les autres sont disponibles pour un usage futur. Notre RIR les attribue de la manière suivante :

— Transitaire 1 : 64496

— Transitaire 2 : 64498

— Transitaire 3 : 65537

— RIR : 64500

— Op. PA Multi : 65540

Concernant l'adressage IP, j'ai décidé de réaliser une maquette uniquement en IPv4. Ce choix s'explique par le fait que la suite logicielle permettant de créer une CA RPKI ne permettait pas de créer des objets cryptographiques liés à des préfixes v6 lorsque j'ai commencé cette maquette. J'ai décidé de choisir mes préfixes dans le préfixe réservé à l'IANA pour la documentation et les tests : 198.18.0.0/15.

On considère que le RIR de la maquette a reçu l'allocation suivante : 198.18.0.0/18. Le reste est disponible pour des allocations futures. Le RIR attribue un unique /21 à tous ses membres :

— RIR : 198.18.0.0/21

— Transitaire 1 : 198.18.16.0/21

— Transitaire 2 : 198.18.32.0/21

— Transitaire 3 : 198.18.48.0/21

Chaque demandeur découpera son /21 en plusieurs /23. Le premier est destiné aux infrastructures (cache validateur, pont de publication mais ça pourrait être site web, ...). Le dernier /23 est destiné aux interconnexions. Les autres sont dédiés aux sous allocations (des /24) ou libres pour un usage futur (expansion, ...).

En pratique, cela donne le découpage suivant : RIR : 198.18.0.0/21

— 198.18.0.0/23 : infrastructures

— 198.18.2.0/23 et 198.18.4.0/23 : libres

— 198.18.6.0/23 : interconnexions

Transitaire 2 : 198.18.32.0/21

— 198.18.32.0/23 : infrastructures

— 198.18.34.0/23 et 198.18.36.0/23 : sous-allocations :

  — 198.18.34.0/24 : Op. PA Single

  — 198.18.35.0/24 : Op. PA Multi

— 198.18.38.0/23 : interconnexions

Transitaire 1 et 3 sont semblables à Transitaire 2.

Les préfixes des interconnexions sont des /30. À la réflexion, des /31 auraient été plus judicieux. Dans la vraie vie, dans le cadre d'un GIX, le préfixe d'interconnexion est fourni par le GIX lui-même. Dans le cadre d'un accord transitaire-client, le transitaire fournit généralement le préfixe d'interconnexion (/30 ou /31). Dans le cadre d'une interconnexion privée, les parties se mettent d'accord sur l'adressage. J'ai choisi de respecter ces pratiques : mes transitaires fourniront les préfixes à leurs clients. Entre deux transitaires, on considérera qu'il y a eu un accord entre les deux AS.

Le nom de chaque machine sera dans le fichier /etc/hosts de toutes les machines de la maquette. Utilisation du TLD spécial test. Forme des noms : machine.ASN.test. Exemple : r1.64496.test. Le nom associé aux routeurs pointe toujours sur l'interface interne. Exemple : 198.18.1.254 pour r1.64500.test.

6.2.3 RPKI

La hiérarchie RPKI sera la suivante :

```
certificat racine du RIR
        certificat d'usage du RIR
                objets cryptographiques du RIR

                certificat du transitaire 2
                        objets cryptographiques du transitaire 2
                        certificat Op. PA Multi
                                objets cryptographiques Op. PA Multi

                certificat du transitaire 3
                        objets cryptographiques du transitaire 3
```

6.2.4 Fausses annonces

Pour illustrer le gain de sécurité apporté par RPKI+ROA ainsi que l'influence des politiques de validation sur cette sécurité, l'AS 65540 annoncera 198.18.0.0/21 (préfixe alloué au RIR, détournement de préfix) et l'AS 65537 annoncera 198.18.32.0/23 (sous-préfixe d'un préfixe attribué à Transitaire 2, détournement d'un sous-préfixe).

Comment j'ai choisi ces préfixes et leurs usurpateurs :

— 198.18.0.0/21 est un bloc important étant donné qu'il contient l'instance de publication du RIR. Il est donc le préfixe le plus intéressant à attaquer de toute la maquette. Choisir OpPAMulti comme usurpateur permet de montrer que participer à la RPKI et faire de la validation ne protège pas de toutes les erreurs.

— Ici, les choix tiennent compte du fait qu'on n'utilise toujours pas Transitaire 1 (AS "témoin"), on ne réutilise pas OpPaMulti. On ne veut pas paralyser les caches validateurs, ce qui exclu RIR et Transitaire 2. Il ne reste donc que Transitaire 3 et le préfixe de Transitaire 2.

6.2.5 Logiciels

Voici la liste des logiciels que j'utiliserai pour construire ma maquette :

— Pour exécuter chaque machine : des LinuX Containers avec Debian Unstable. Les solutions de virtualisation classiques (VirtualBox, KVM) m'ont paru trop lourdes par rapport aux capacités de ma machine et des besoins de la maquette. UML (avec Netkit ou Marionnet en interface) pose problème, surtout au niveau du noyau, pour avoir un système récent nécessaire notamment pour l'AC.

— Pour créer l'autorité de certification du RIR : RPKI tools de rpki.net, seule implémentation libre et complète.

— Caches validateurs : RPKI-validator et rcynic, seules implémentations libres potables seront déployés sur les 2 caches validateurs. RPKi-validator rencontre parfois des erreurs incompréhensibles donc avoir une autre implémentation en marche permet d'éviter de bloquer la maquette. L'AS 64500 utilisera uniquement rpki-validator pour valider. l'AS 64498 et 65540 utiliseront uniquement rcynic.

— Routeur : Quagga et le patch BGP-SRx, seule implémentation libre et complète.

6.2.6 Dernières précisions

J'aurai voulu de la redondance :

— basculer automatiquement entre les deux implémentations de cache validateur ;

— qu'un AS utilise son cache validateur local mais aussi celui de son transitaire en backup comme recommandé par les RFC.

mais l'implémentation BGP-SRx supporte uniquement l'usage d'un seul cache validateur dans un même temps.

J'aurai voulu que la communication entre un cache validateur et un routeur soit sécurisée mais Quagga SRX prend en charge uniquement les connexions TCP nues. J'aurai pu utiliser SSH et le port forwarding mais j'aurai été loin d'un déploiement réalise.

J'ai tenté de faire une maquette IPv4/IPv6. Voici les conclusions :

— La création des certificats et des ROA avec des entrées IPv6 ne pose aucun problème.

— Concernant l'adressage, le RIR de la maquette a reçu 2001 :db8 : :/32 (préfixe de documentation) et alloue des /48 à Transitaire 1, Transitaire 2, Transitaire 3 et à lui-même. La logique du découpage est identique à celle utilisée pour IPv4 sauf qu'il ne s'agit plus de /23 mais de /56 (ce choix est discutable). Transitaire 2 alloue un /56 à OpPaSingle et à OpPaMulti. Les interconnexions se font sur des préfixes /127.

— La montée de sessions BGP sur IPv6 ainsi que l'échange de préfixes IPv6 avec Quagga ne posent pas de problème.

— Les caches validateurs valident bien les ROA "v6" et les distribuent bien en RTR (j'ai vérifié avec rtrclient).

— Quagga ne récupère pas l'état de validation. Puisque les caches validateurs fonctionnent, il s'agit d'un problème interne à BGP-SRx. La documentation n'indique rien au sujet d'IPv6. J'ai envoyé un mail aux développeurs : pas de réponse à ce jour.

6.3 Résultats

6.3.1 Mise en marche

D'abord, on observe que la RPKI est fonctionnelle et que les caches-validateurs valident tous les objets cryptographiques de la maquette :

Les 4 avertissements proviennent du fait que les CA tools de rpki.net génèrent les Manifest d'une manière permise par les RFC mais non recommandée. Avec les 15 objets validés, ils forment l'intégralité des objets de ma RPKI : certificats, listes de révocation, ROAs et Manifest.

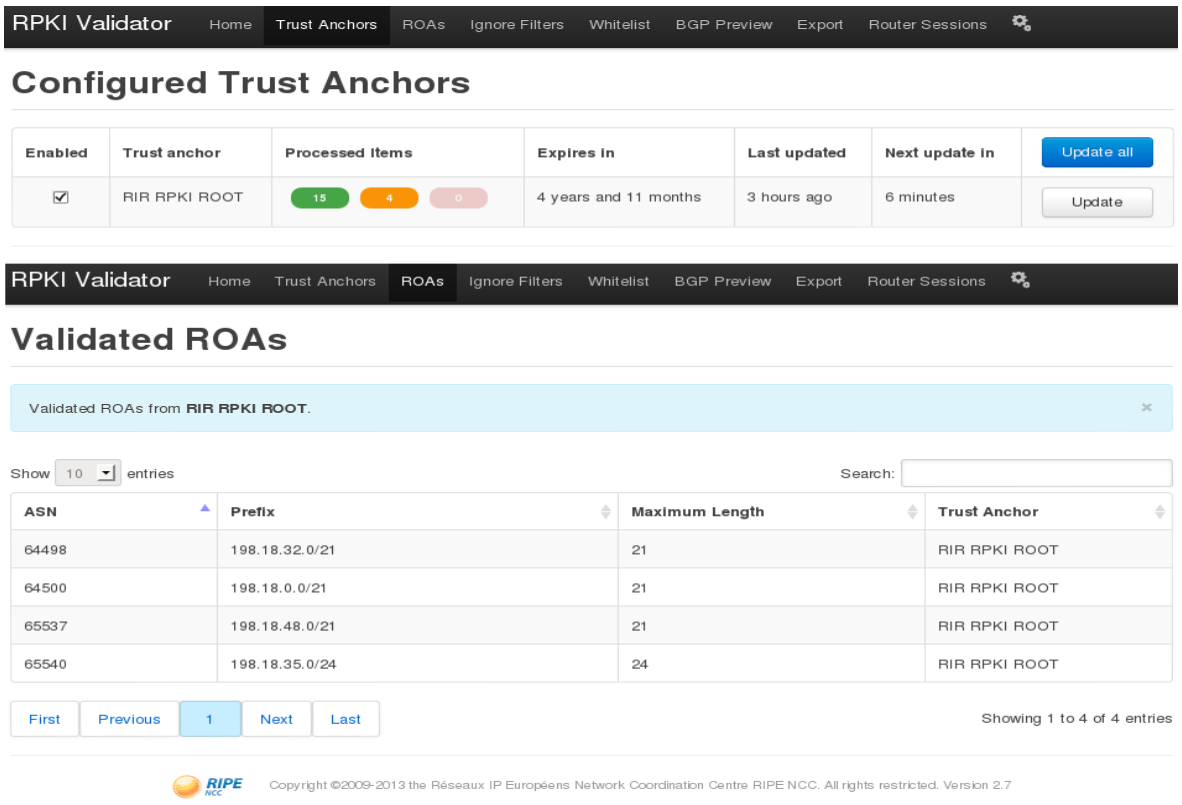

FIGURE 6 – L'instance RPKI-validator de cache.rir.test valide tous les objets cryptographiques.

De plus, nous observons que l'état de la validation est remonté jusqu'à nos routeurs. Ici, sur r1.64498.test

```
   Ident     SRxVal SRxLP Status Network         Next Hop      Metric  LocPrf Weight Path
# Ici, nous avons une route invalide : 65540 n'est pas l'AS indiqué dans le ROA
*  B29D4BDB i(i,-)              198.18.0.0/21   198.18.38.10  0        0 65540 i

# Ces deux routes ont les mêmes caractéristiques (préférence, métrique,
# état de validation = valide) donc celle ayant le plus court chemin d'AS l'emporte
*  DECF90D7 v(v,-)              198.18.0.0/21   198.18.54.1            0 65537 65540 i
*  18181D3D v(v,-)                              198.18.22.9            0 64496 64500 i
*> 8C32750E v(v,-)                              198.18.38.2  0         0 64500 i

# Transitaire 1 n'a pas crée d'objets cryptographiques donc l'état de validation est
# inconnu. L'état indéfini (= ?) pour la même annonce est un bug de Quagga-SRx
*  20E4987E u(u,-)              198.18.16.0/21 198.18.54.1            0 65537 64496 i
*> CB32E97D ?(?,-)                            198.18.22.9  0         0 64496 i
```

```
# La validation RPKI+ROA n'est jamais effectuée pour les préfixes locaux
*> -------- ?(?,-)              198.18.32.0/21   0.0.0.0    0         32768 i
```

Enfin, on obverse que les fausses annonces sont aussi propagées (ici, sur r1.64496.test) :

```
*  -------- ?(?,-)              198.18.0.0/21  198.18.22.2            0 65537 65540 i
*  -------- ?(?,-)              198.18.32.0/23 198.18.22.10           0 64498 65537 i
*> -------- ?(?,-)                             198.18.22.2  0         0 65537 i
```

6.3.2 Apports de RPKI+ROA

Comparons la table de routage BGP de routeurs qui ne font pas de validation RPKI+ROA à celle d'un routeur qui fait de la validation et plus particulièrement les entrées relatives à la fausse annonce pour 198.18.0.0/21 :

```
T1>show ip bgp
   Ident    SRxVal SRxLP Status Network        Next Hop     Metric  LocPrf Weight Path
*  -------- ?(?,-)              198.18.0.0/21  198.18.22.2            0 65537 65540 i
*  -------- ?(?,-)                             198.18.22.10           0 64498 64500 i
*> -------- ?(?,-)                             198.18.22.6  0         0 64500 i
```

On remarque que le routeur BGP de Transitaire 1, qui ne fait pas de validation, accepte la fausse annonce. Les critères de comparaison étant tous identiques, Il sélectionne la bonne route simplement car le chemin d'AS est plus court.

```
T3>show ip bgp
   Ident    SRxVal SRxLP Status Network        Next Hop     Metric  LocPrf Weight Path
*> -------- ?(?,-)              198.18.0.0/21  198.18.54.6  0         0 65540 i
*  -------- ?(?,-)                             198.18.54.2            0 64498 64500 i
*  -------- ?(?,-)                             198.18.22.1            0 64496 64500 i
```

Le raisonnement précédent se confirme sur le routeur BGP du transitaire 3 : cette fois-ci le chemin d'AS de l'annonce légitime est plus long donc la fausse annonce est sélectionnée.

```
T2>show ip bgp
*  B29D4BDB i(i,-)             198.18.0.0/21  198.18.38.10  0         0 65540 i
*  DECF90D7 i(i,-)                            198.18.54.1            0 65537 65540 i
*  18181D3D v(v,-)                            198.18.22.9            0 64496 64500 i
*> 8C32750E v(v,-)                            198.18.38.2   0         0 64500 i
```

Sur un routeur BGP qui fait de la validation RPKI+ROA, ici le routeur de Transitaire 2, on observe que la fausse annonce est marquée comme invalide. Comme il existe une autre route qui, de plus, profite d'un crédit plus élevé (elle est valide), la politique « prefer-valid » fera que la route invalide ne sera pas sélectionnée (sauf si l'AS légitime tombe, bien sûr car il n'y aura plus alors de routes alternatives). Toutes les annonces ne se valent plus : l'annonce sélectionnée l'est car elle a un chemin d'AS plus court mais surtout car elle est valide.

6.3.3 Limites de « prefer-valid »

Comparons maintenant les tables de routage BGP pour le sous-préfixe usurpé, 198.18.32.0/23.

```
T1>show ip bgp
   Ident    SRxVal SRxLP Status Network        Next Hop     Metric  LocPrf Weight Path
*  -------- ?(?,-)              198.18.32.0/23 198.18.22.10           0 64498 65537 i
*> -------- ?(?,-)                             198.18.22.2  0         0 65537 i
```

On remarque que le routeur BGP de Transitaire 1, qui ne fait pas de validation, accepte la fausse annonce. Les critères de comparaison étant tous identiques, Il sélectionne une des routes sur le critère du chemin d'AS le plus court.

```
RIR>show ip bgp
# Deux états de validation pour une même route est un bug récurrent et aléatoire de
# BGP SRx.
*  58ECE0C6 u(u,-)              198.18.32.0/23 198.18.22.5            0 64496 65537 i
*> 58ECE0C6 i(i,-)                             198.18.38.1            0 64498 65537 i
```

On retrouve ce comportement sur le routeur du RIR qui pourtant fait de la validation. C'est la conséquence logique de l'algorithme « prefer-valid » : il n'y a pas d'autres routes ayant un état de validation préféré (valide, inconnu) donc on sélectionne la route invalide afin de conserver la connectivité au réseau concerné.

Le seul moyen d'éviter cette situation est d'ajouter, à la politique « prefer-valid », une politique « ignore-invalid ». Exemple sur OpPaMulti :

```
*> 964E5B7E v(v,-)              198.18.32.0/21 198.18.38.9  0         0 64498 i
*  58ECE0C6 i(i,-)          I   198.18.32.0/23 198.18.38.9            0 64498 65537 i
*  7779E423 i(i,-)          I                  198.18.54.5  0         0 65537 i
```

Les annonces concernant le sous-préfixe sont ignorées (« I ») et la route moins spécifique est préférée.

On arrive à une situation où l'on se protège des attaques malveillantes mais où l'on n'est plus tolérant aux erreurs humaines dans la RPKI : si un ROA ne correspond plus à la topologie réelle ou si toute autre erreur survient, le ou les préfixes concernés seront ignorés. Cela pose un véritable problème : comment justifier l'implication dont les opérateurs vont devoir faire preuve pour déployer RPKI+ROA tout ça pour arriver à un dilemme "j'ignore les annonces invalides, je peux (et même je vais) perdre de la connectivité à certains réseaux, je n'ignore pas les annonces invalides et je suis vulnérable" ?

7 Conclusion

Dans ce rapport, j'ai présenté les lacunes du routage sur Internet du point de vue de sa sécurité ainsi que RPKI+ROA, seul système normalisé et en cours de déploiement permettant de valider l'origine des annonces BGP, première étape d'un routage entièrement sécurisé (validation de l'origine mais aussi du chemin) ainsi que les limites de ce mécanisme.

Ce projet d'étude m'a permis de revoir les bases et de parfaire les connaissances que j'avais du routage inter-domaines. Il m'a également permis de découvrir et de mettre en œuvre, en détail, RPKI+ROA, un système de sécurité qui sera peut-être amené, dans la dizaine d'années à venir, à être un incontournable dès que l'on raccordera un nouveau réseau au reste d'Internet.

Il est encore trop tôt pour présager du succès du déploiement de RPKI+ROA.

Actuellement, environ 3000 ROA sont présents dans le système de dépôts et permettent de vérifier les annonces d'environ 4000 préfixes soit très peu en comparaison des 467000 préfixes annoncés actuellement (v4 + v6). Concernant la validation des annonces avec RPKI+ROA, personne ne le fait en production ou du moins, personne ne le revendique publiquement et aucune étude n'existe sur le sujet, à ma connaissance.

D'un côté, plusieurs solutions de sécurité n'ont jamais rencontré le succès : PGP, IPSec, ... et les opérateurs de réseaux ont des déploiements plus urgents comme IPv6. D'un autre côté, l'implication des RIR et de nouveaux RFC qui viennent combler les manques actuels de RPKI+ROA, rendent le système plus crédible et accessible aux opérateurs réseaux.

En plus de la validation de l'origine qu'apporte RPKI+ROA, il faudra valider les chemins afin arriver à une solution complète de sécurisation du routage sur Internet, un BGPSec. Cette problématique n'est pas triviale et la réflexion est déjà en cours au sein du groupe de travail sidr de l'IETF.

8 Références

[BORTZMEYER(2012)] Stéphane BORTZMEYER. La longue marche de la sécurité du routage internet : une étape importante, rpki+roa, février 2012. URL `http://www.bortzmeyer.org/securite-routage-bgp-rpki-roa.html`. Présentation de RPKI+ROA sur le blog de Stéphane Bortzmeyer. Tous les commentaires des RFC relatifs à RPKI+ROA par le même auteur cités dans cette page web m'ont aidés.

[CHARBONNIER(2007)] Laurent CHARBONNIER. *Evaluation de la sécurité des réseaux privés virtuels sur MPLS*. PhD thesis, Université du Québec, 2007. URL `http://espace.etsmtl.ca/239/4/CHARBONNIER_Laurent-web.pdf`.

[Charles Lynn(2003)] Karen Seo Charles Lynn, Joanne Mikkelson. Secure bgp (s-bgp). Internet draft, IETF, juin 2003. URL `http://www.ir.bbn.com/sbgp/draft-clynn-s-bgp-protocol-01.txt`.

[DEF(2008)] *Stealing the Internet*, août 2008. DEFCON 16. URL `https://media.defcon.org/DEF%20CON%2016/DEF%20CON%2016%20video/DEF%20CON%2016%20Hacking%20Conference%20Presentation%20By%20Kapela%20and%20Pilosov%20-%20Stealing%20the%20Internet%20-%20Video.m4v`. Enregistrement vidéo de la présentation de Kapela et Pilosov à la DEFCON 16. Vérifié le 13/01/2013.

[Fares(2004)] Joseph E. Fares. *Routage inter-domaine - Sécurité et enjeux*. PhD thesis, Université Libanaise, université Saint-Joseph, 2004. URL `http://www.lb.refer.org/memoires/458635JosephFARES.doc`. Travail universitaire sur BGP et les attaques sur BGP.

[François CONTAT(2012)] Guillaume VALADON François CONTAT, Sarah NATAF. Influence des bonnes pratiques sur les incidents bgp. 2012. URL `https://www.sstic.org/media/SSTIC2012/SSTIC-actes/influence_des_bonnes_pratiques_sur_les_incidents_b/SSTIC2012-Article-influence_des_bonnes_pratiques_sur_les_incidents_bgp-contat_valadon_nataf_2.pdf`.

[FRn(2012)] *Sécurité des annonces BGP : RPKI+ROA sauveront-elles le monde ?*, juin 2012. FRnOG 19. URL `http://www.dailymotion.com/video/xtp7r9_frnog-19-securite-des-annonces-bgp-rpki-roa-sauveront-elles-le-monde-stephane-bortzmeyer-tech`. Enregistrement vidéo de la présentation RPKI+ROA de Stéphane BORTZMEYER à la réunion FRnOG 19. Vérifié le 13/01/2013.

[Gallais(2012)] Antoine Gallais. *Sécurité des réseaux - Attaques réseaux*, 2012. Cours Sécurité des réseaux dispensé aux M1 RISE - Université de Strasbourg.

[Hitesh Ballani(2007)] Paul Francis et Xinyang Zhang Hitesh Ballani. A study of prefix hijacking and interception in the internet. In *slides*, août 2007. URL `http://www.soi.wide.ad.jp/project/sigcomm2007/pdf/sig81.pdf`. Présentation d'un détournement de préfixe lors de la SIGCOMM 2007.

[Horn(2009)] Christian Horn. Understanding ip prefix hijacking and its detection. Technical report, juin 2009. URL https://www.net.t-labs.tu-berlin.de/teaching/ss09/IR_seminar/talks/prefix_hijacking_horn.handout.pdf. Détournement de préfixe et détection.

[J. Gersch(2012)] C. Olschanowsky et L. Zhang J. Gersch, D. Massey. Dns resource records for bgp routing data. Internet draft, IETF, août 2012. URL https://tools.ietf.org/id/draft-gersch-grow-revdns-bgp-01.txt.

[Kent(?)] Stephen T. Kent. Securing the border gateway protocol. Technical report, ? ? URL http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_6-3/securing_bgp_s-bgp.html.

[Murphy(2006)] S. Murphy. BGP Security Vulnerabilities Analysis. RFC 4272, RFC Éditor, janvier 2006. URL http://www.rfc-editor.org/rfc/rfc4272.txt.

[Osterweil et al.(2012)Osterweil, Manderson, White, and McPherson] Eric Osterweil, Terry Manderson, Russ White, and Danny McPherson. Sizing estimates for a fully deployed rpki. Technical Report 1120005 version 2, 2012. URL http://techreports.verisignlabs.com/docs/tr-1120005-2.pdf.

[P. C. VAN OORSCHOT(2007)] EVANGELOS KRANAKIS P. C. VAN OORSCHOT, TAO WAN. On interdomain routing security and pretty secure bgp (psbgp). Technical report, Université de Carleton, juillet 2007. URL http://people.scs.carleton.ca/~paulv/papers/tissec-july07.pdf.

[Rexford(2006)] Josh Karlin Stephanie Forrest Jennifer Rexford. Pretty good bgp : Improving bgp by cautiously adopting routes. Technical report, Université de New-Mexico / Université Princeton, 2006. URL http://www.cs.unm.edu/~treport/tr/06-06/pgbgp3.pdf.

[WG(2012-2013)] SIDR WG. Secure Inter-Domain Routing (sidr) Documents. Technical report, RFC Éditor, 2012-2013. URL https://datatracker.ietf.org/wg/sidr/. Documents du groupe de travail SIDR à l'IETF.

[White(?)] R. White. Securing bgp through secure origin bgp. Technical report, ? ? URL http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_6-3/securing_bgp_sobgp.html.

[White(2003)] R. White. Deployment considerations for secure origin bgp (sobgp). Internet draft, IETF, juin 2003. URL https://tools.ietf.org/html/draft-white-sobgp-bgp-deployment-01.

[White(2005)] R. White. Architecture and deployment considerations for secure origin bgp (sobgp). Internet draft, IETF, juin 2005. URL https://tools.ietf.org/html/draft-white-sobgp-architecture-02.

A Annexes

A.1 Création de la maquette

Note : tous les scripts et fichiers de configuration mentionnés dans ce guide sont disponibles à l'adresse suivante : `http://www;guiguishow.info/wp-content/uploads/2013/09/RPKI-ROA/Maquette/Configuration-maquette_RPKI-ROA.tar.bz2`.

A.1.1 Création des conteneurs LXC

Pour créer les 9 conteneurs de ma maquette, j'ai décidé d'écrire et d'utiliser un petit script (lxc-create-all.sh) qui permet de semi-automatiser la création. Il suffit de l'exécuter et de paramétrer la création de chaque conteneur (normalement les preseed file permettent de faire cela automatiquement mais cela n'a pas totalement fonctionné donc j'ai décidé de tout faire manuellement [31]) :

- Pas de preseed file
- Debian Unstable
- 64-bit
- Pas besoin du dépôt experimental
- Miroir : le plus proche donc ftp ://ftp.u-strasbg.fr/debian/
- Uniquement main
- Les packages à installer dépendent de l'usage du conteneur :
  - Pour l'AC (P1-64500) : less iputils-ping traceroute iptables wget nano rsyslog bash-completion telnet p7zip-full ethtool dnsutils python2.7-dev python-setuptools python-lxml libxml2-utils mysql-client mysql-server python-mysqldb python-django python-vobject python-yaml xsltproc rrdtool libapache2-mod-wsgi chrootuid gcc build-essential automake autoconf make subversion rsync bsdmainutils . Si vous ne voulez pas utiliser l'interface web, vous pouvez ne pas installer python-django, python-vobject et libapache2-mod-wsgi.
  - Pour les caches-validateurs (C1-64498 et C1-64500) : less iputils-ping traceroute iptables wget nano rsyslog bash-completion telnet p7zip-full ethtool dnsutils openjdk-6-jre openjdk-7-jre python2.7-dev python-setuptools python-lxml libxml2-utils python-yaml xsltproc rrdtool chrootuid gcc build-essential automake autoconf make subversion rsync bsdmainutils xinetd
  - Pour les routeurs BGP : less iputils-ping traceroute iptables wget nano rsyslog bash-completion telnet p7zip-full ethtool dnsutils automake autoconf libtool texinfo gawk libncurses5 libncurses5-dev readline-common libconfig9 libconfig++9 libconfig-dev libreadline5 libreadline6 libreadline-dev make gcc build-essential

31. Depuis la finalisation de ce travail, j'ai créé un preseed-file potable. Voir : `http://www.guiguishow.info/2013/09/14/maquetter-des-reseaux-avec-lxc/`.

— Pour le dernier routeur sans BGP : less iputils-ping traceroute iptables wget nano rsyslog bash-completion telnet p7zip-full ethtool dnsutils

— Choisir un mot de passe

— Recommencer cette procédure pour les conteneurs restants.

A.1.2 Configuration du réseau

Pour chaque conteneur, il faut rajouter quelques lignes à la fin du fichier « config » qui se trouve à la racine du conteneur. Exemple pour R1-OPAS :

```
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.name = eth0
lxc.network.ipv4 = 198.18.38.6/30
lxc.network.ipv6 = 2001:db8:4498:ff00::5/127
lxc.network.veth.pair = veth_R1_OPAS_1
lxc.network.hwaddr = 02:00:00:00:00:19

lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.name = eth1
lxc.network.ipv4 = 198.18.34.254/24
lxc.network.ipv6 = 2001:db8:4498:01ff:ffff:ffff:ffff:ffff/56
lxc.network.veth.pair = veth_R1_OPAS_2
lxc.network.hwaddr = 02:00:00:00:00:27
```

Pour chaque conteneur, il faut modifier : l'adresse IPv4/IPv6, l'adresse MAC de l'interface, le nom de l'interface sur l'hôte physique (network.veth.pair) et, si plusieurs interfaces, incrémenter lxc.network.name. Tous ces paramètres doivent être propres à un conteneur.

Pour les conteneurs C1-64498, C1-64500 et P1-64500, on précisera aussi la passerelle de sortie (le R1 du sous-réseau) avec « lxc.network.ipv4.gateway » et « lxc.network.ipv6.gateway ». Exemple pour C1-64498 :

```
lxc.network.ipv4.gateway = 198.18.33.254
lxc.network.ipv6.gateway = 2001:db8:4498:00ff:ffff:ffff:ffff:ffff
```

A.1.3 Premier lancement

Le script « lxc-start.sh » permet d'exécuter les tâches nécessaires au bon fonctionnement de la maquette : créer le bridge, monter les cgroups dont dépend LXC et configurer le pare-feu de l'hôte pour laisser passer le trafic sur le bridge puis de démarrer toutes les machines.

Le script « lxc-console-all.sh » permet d'ouvrir un terminal sur chaque conteneur.

On profitera d'avoir un terminal ouvert sur tous les conteneurs pour supprimer exim4 qui provoque parfois des problèmes lors du démarrage des conteneurs : apt-get autoremove -y –purge exim4 exim4-base exim4-config && apt-get clean

On activera le forward IP sur tous les routeurs sauf R1-OPAS et R1-65540 où cela n'est pas nécessaire en décommentant les lignes suivantes dans /etc/sysctl.conf :

```
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

Enfin, sur chaque conteneur, on déploiera un fichier /etc/hosts qui contient l'association entre le nom de nos machines et leurs adresses.

```
198.18.32.1 cache.64498.test
198.18.0.1 cache.64500.test
198.18.0.2 pub.64500.test
198.18.17.254 r1.64496.test
198.18.33.254 r1.64498.test
198.18.1.254 r1.64500.test
198.18.49.254 r1.65537.test
198.18.35.254 r1.65540.test
198.18.34.254 r1.opas.test
2001:db8:4498::1 cache.64498.test
2001:db8:4500::2 cache.64500.test
2001:db8:4500::1 pub.64500.test
2001:db8:4496:00ff:ffff:ffff:ffff:ffff  r1.64496.test
2001:db8:4498:00ff:ffff:ffff:ffff:ffff  r1.64498.test
2001:db8:4500:00ff:ffff:ffff:ffff:ffff  r1.64500.test
2001:db8:5537:00ff:ffff:ffff:ffff:ffff  r1.65537.test
2001:db8:4498:02ff:ffff:ffff:ffff:ffff  r1.65540.test
2001:db8:4498:01ff:ffff:ffff:ffff:ffff  r1.opas.test
```

A.1.4 Installer l'AC (rpki tools) sur P1-64500

1. Connecter le conteneur au reste d'Internet. Voir plus loin en annexe pour la procédure à suivre.
2. Si vous voulez utiliser l'interface web, installez python-django-south : easy_install South==0.7.6 . Les dépôts Debian ne contiennent pas encore la version requise (0.7.6).
3. Récupérer les sources : svn checkout http ://subvert-rpki.hactrn.net/trunk/
4. Compilation : cd trunk && ./configure && make && make install (appuyez sur entrée lors de la demande de votre mot de passe mysql : il n'y en a pas par défaut)
5. Copier les fichiers rpki.conf et rsyncd.conf (fournis) dans P1-64500/rootfs/usr/local/etc/ depuis l'hôte. Ils permettent de configurer les différentes instances de l'AC (AC racine, AC standard, instance de publication).
6. Créer la hiérarchie de l'instance de publication : mkdir -p /var/rpki/publication,publication.root
7. Générer le certificat racine. Note : Debian ne propose pas OpenSSL avec les extensions nécessaires (RFC 3779). Les RPKI tools compilent une version avec ces extensions.
   (a) Générer une paire de clés : openssl genrsa -out root.key 2048
   (b) Générer la requête de certification : openssl req -new -config root.conf -out root.req -key root.key
   (c) Signer le certificat : ./trunk/openssl/openssl-1.0.1e/apps/openssl x509 -req -sha256 -signkey root.key -in root.req -outform DER -out root.cer -extfile root.conf -extensions x509v3_extensions -days 1825

   root.conf est fourni. Vous pouvez le modifier pour choisir les ASN/préfixes distribués au RIR, le nom de votre RIR, ...
8. Copier root.key dans /usr/local/share/rpki/ et root.cer dans le point de publication de la racine tel qu'il est spécifié dans rpki.conf (dans notre cas : /var/rpki/publication.root/) : cp root.key /usr/local/share/rpki/ && cp root.cer /var/rpki/publication.root/
9. rpki-sql-setup –drop-and-create
10. rpkic initialize
11. rpki-start-servers
12. rpkic configure_publication_client RIR.RIR.repository-request.xml && rpkic configure_repository RIR.repository-response.xml
13. Création de la hiérarchie RPKI (création des LIR/opérateurs, délégation puis création des ROAs) :

```
rpkic
select_identity transitaire2
```

```
initialize
select_identity RIR
configure_child transitaire2.identity.xml
select_identity transitaire2
configure_parent RIR.transitaire2.parent-response.xml
select_identity RIR
configure_publication_client transitaire2.RIR.repository-request.xml
select_identity transitaire2
configure_repository RIR.transitaire2.repository-response.xml

select_identity opam
initialize
select_identity transitaire2
configure_child opam.identity.xml
select_identity opam
configure_parent transitaire2.opam.parent-response.xml
select_identity transitaire2
configure_publication_client opam.transitaire2.repository-request.xml
select_identity opam
configure_repository RIR.transitaire2.opam.repository-response.xml

select_identity transitaire3
initialize
select_identity RIR
configure_child transitaire3.identity.xml
select_identity transitaire3
configure_parent RIR.transitaire3.parent-response.xml
select_identity RIR
configure_publication_client transitaire3.RIR.repository-request.xml
select_identity transitaire3
configure_repository RIR.transitaire3.repository-response.xml



select_identity RIR
load_prefixes transitaires_prefixes_list.csv
load_asns transitaires_asns_list.csv
```

```
load_roa_requests rir_roa_list.csv

select_identity transitaire3
load_roa_requests transitaire3_roa_list.csv

select_identity transitaire2
load_roa_requests transitaire2_roa_list.csv

select_identity transitaire2
load_prefixes opam_prefixes_list.csv
load_asns opam_asns_list.csv

select_identity opam
load_roa_requests opam_roa_list.csv
```

14. rsync –daemon –config=/usr/local/etc/rsyncd.conf
15. Enfin, il faut faire démarrer la suite logicielle automatiquement à chaque démarrage de P1-64500.
    (a) On crée une ébauche de script d'init (exemple : /etc/init.d/rpkid) :

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:          rpkid
# Required-Start:    mysql
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# X-Interactive:     false
# Short-Description: Start/stop rpkid
### END INIT INFO

/usr/local/sbin/rpki-start-servers &
/bin/rm /var/run/rsyncd.pid
/usr/bin/rsync rsync --daemon  --config=/usr/local/etc/rsyncd.conf &
```

    (b) chmod +x /etc/init.d/rpkid && update-rc.d rpkid defaults

Si vous voulez utiliser l'interface web, il y a quelques étapes en plus (configurer Apache et Django) :

1. cp /usr/local/etc/rpki/apache.conf /etc/apache2/conf.d/
2. a2enmod ssl
3. make-ssl-cert generate-default-snakeoil
4. a2ensite default-ssl
5. service apache2 restart
6. rpki-manage syncdb (quand demandé, choisissez de créer un utilisateur) && rpki-manage migrate app

A.1.5 Installer le cache-validateur RPKI-Validator

1. Récupérer RPKI-Validator sur le site du RIPE : `https://www.ripe.net/lir-services/resource-manage certification/tools-and-resources`.
2. 7z x rpki-validator-app-2.7-bin.zip
3. mv rpki-validator-app-2.7 /usr/local/sbin/
4. rm /usr/local/sbin/rpki-validator-app-2.7/conf/tal/*
5. Créer le fichier TAL.
   (a) Récupérer la clé publique du certificat racine sur P1-64500 : ./trunk/openssl/openssl-1.0.0i/apps/openssl x509 -inform DER -noout -pubkey -in root.cer.
   (b) Créer un fichier, /usr/local/sbin/rpki-validator-app-2.7/conf/tal/pub-64500.tal, avec le contenu suivant :

   ```
   ca.name = RIR RPKI ROOT
   certificate.location = rsync://pub.64500.test/root/root.cer
   public.key.info = [clé publique sur une seule ligne]
   prefetch.uris = rsync://pub.64500.test/rpki/
   ```

6. Il ne reste plus qu'à créer un script pour lancer le cache-validateur automatiquement au démarrage du conteneur. Là encore, on utilise un script d'init avec le contenu suivant :

   ```
   #!/bin/bash

   ### BEGIN INIT INFO
   # Provides:          rpki-validator
   # Required-Start:    routing
   # Required-Stop:
   # Default-Start:     2 3 4 5
   ```

```
# Default-Stop:      0 1 6
# X-Interactive:     false
# Short-Description: Start/stop rpki-validator
### END INIT INFO

cd /usr/local/sbin/rpki-validator-app-2.7/
./bin/rpki-validator &
```

A.1.6 Installer le cache-validateur rcynic

1. Connecter le conteneur au reste d'Internet
2. Récupérer les sources : svn checkout http ://subvert-rpki.hactrn.net/trunk/
3. Compilation : cd trunk && ./configure –disable-ca-tools && make && make install
4. Créer un script bash, /var/rcynic/validation.sh qui lancera la validation. Il doit avoir le contenu suivant :

```
#!/bin/sh -
/usr/local/bin/rcynic -c /var/rcynic/etc/rcynic.conf || exit
#/var/rcynic/bin/rcynic-html /var/rcynic/data/rcynic.xml /var/www/rcynic
cd /var/rcynic/rpki-rtr
/usr/local/bin/rtr-origin --cronjob /var/rcynic/data/authenticated
```

5. Rendre ce script exécutable : chmod +x /var/rcynic/script.cron
6. Le faire exécuter par cron à intervalle régulier (ici 4H pour être synchronisé avec RPKI-Validator) : 0 */4 * * * /var/rcynic/script.cron dans crontab -e
7. Modifier le fichier /var/rcynic/etc/rcynic.conf pour qu'il corresponde au contenu suivant :

```
[rcynic]
rsync-program           = /usr/bin/rsync
authenticated           = /var/rcynic/data/authenticated
unauthenticated         = /var/rcynic/data/unauthenticated
lockfile                = /var/rcynic/data/lock
xml-summary             = /var/rcynic/data/rcynic.xml
jitter                  = 600
use-syslog              = false
use-stderr              = true
log-level               = log_debug
```

```
# Using the "Trust Anchor Locator" form:
trust-anchor-locator.1  = /var/rcynic/etc/trust-anchors/rir.tal
```

8. mkdir /var/rcynic/etc/trust-anchors
9. Créer le fichier TAL en /var/rcynic/etc/trust-anchors. Le format utilisé est celui normalisé dans le RFC 6490. La clé publique s'obtient de la même façon que lors de la configuration de RPKI-Validator.

   ```
   rsync://pub.64500.test/root/root.cer
   [clé publique sur une seule ligne]
   ```

10. Il ne reste plus qu'à rendre accessible le cache-validateur via RTR :
    (a) echo "rpki-rtr 42420/tcp" > > /etc/services
    (b) Créer un nouveau fichier décrivant un service xinetd (/etc/xinetd.d/rpki-rtr par exemple) contenant :

    ```
    service rpki-rtr
    {
        type           = UNLISTED
        flags          = IPv4
        socket_type    = stream
        protocol       = tcp
        port           = 43779
        wait           = no
        user           = nobody
        server         = /usr/bin/rtr-origin
        server_args    = --server /var/rcynic/rpki-rtr
    }
    ```

    (c) service xinetd restart

A.1.7 Installer Quagga

J'ai décidé d'utiliser la même version de Quagga pour tous mes routeurs BGP, même pour ceux qui ne font pas de validation RPKI+ROA.

Le script « install-Quagga.sh » permet de compiler BGP-SRx puis de compiler/installer Quagga. Il faut le mettre sur chaque routeur ainsi que le bundle Quagga-SRx récupérable sur le site du NIST (ou sur `http://www.guiguishow.info/wp-content/uploads/2013/09/RPKI-ROA/nist-srx-bundle-0.2.0.tar` vu que le NIST demande un mail + pays + organisation pour récupérer les sources d'un simple logiciel qui se trouve dans le domaine public ...).

Il faut ensuite effectuer la configuration classique de BGPd (pairs, préfixe à annoncer, ...) et de Zebra (interfaces).

Sur R1-64498, ne pas oublier d'ajouter des routes statiques (dans zebra.conf) pour joindre le réseau d'OPAS :

```
ip route 198.18.34.0/24 198.18.38.6
ipv6 route 2001:db8:4498:0100::/56 2001:db8:4498:ff00::5
```

A.1.8 Activer la validation RPKI+ROA

Pour activer la validation sur les routeurs voulus (R1-64500, R1-64498 et 65540 dans mon cas), il faut rajouter quelques lignes dans bgpd.conf. Exemple sur R1-64498 :

```
srx connect localhost 17900
srx display
srx evaluation origin_only
srx keep-window 900
no srx policy ignore-undefined
srx policy prefer-valid
```

Pour lancer le serveur SRx au démarrage des routeurs, il faut ajouter les deux lignes suivantes au script d'init /etc/init.d/routing :

```
/usr/local/bin/srx_server --port 17900 --console.password toor --rpki.host cache.64500.test
--rpki.port 8282 --bgpsec.host noexist.test --bgpsec.port 2365 &
sleep 5
```

A.1.9 Astuce : relier un conteneur à Internet

Sur l'hôte : brctl addif br0 eth0

Dans le conteneur :

```
ip r d default
dhclient -v eth0 ou définir un adressage statique/résolveur DNS
```

A.1.10 Astuce : accéder à un conteneur depuis l'hôte

Il suffit de donner une adresse IP à l'interface br0 de l'hôte. Bien sûr, l'adresse doit être sur le même réseau que le conteneur.

Exemple : pour joindre P1-64500 : ip a a br0 198.18.0.5/23

Glossaire

Autonomous System Un AS correspond, pour simplifier, à une entité administrative qui participe aux opérations qui relevent de la couche 3 du modèle OSI, comme le routage. Les AS sont numérotés de manière unique à l'échelle d'Internet : un AS = un numéro d'AS (ASN, AS Number). En réalité, un AS représente un ensemble de réseaux qui partagent une politique de routage externe commune. Une politique de routage se compose d'un certain nombre de choix qui déterminent le routage local. Il s'agit de choix que l'on ne pourrait pas prendre à l'échelle globale d'Internet. Exemples (les deux premiers sont farfelus, juste pour donner une idée) : nos routeurs seront uniquement de la marque Juniper. Demain, on déploiera IPv6 dans notre cœur de réseau. On péere avec les AS qui correspondent à tels critères et selon telles conditions. Si plusieurs routes sont disponibles pour un même préfixe, on sélectionne la "meilleure" route selon tels critères subjectifs (on ne veut pas passer par tel AS car on a des doutes le concernant, on veut passer en priorité par tel AS car on a un contrat de transit moins cher, si l'origine est X, alors on choisira telle autre route, ...). Un opérateur peut avoir plusieurs AS de même qu'un AS peut être assigné à un conglomérat d'opérateurs. Par exemple : Level 3, gros fournisseur de transit, dispose de plusieurs AS (3549, 6395, 8043, 3356, ...). Cela fait sens quand une même entité (ici Level 3) a des politiques de routage différentes en fonction, par exemple, de la région (Europe, Amérique du Nord, ...). RENATER aussi fait cela à l'échelle nationale (RENATER Limousin, Martinique, Rhone-Alpes, ...) Cela peut aussi se produire lors de fusion-acquisition. Exemple : SFR rachète la fusion-acquisition Neuf-Cegetel et récupère donc l'AS de Cegetel (AS 8228) et l'AS de Neuf (AS 15557) en plus du sien (AS 39079).

Default-Free Zone Le terme default-free zone (DFZ) désigne l'ensemble des routeurs qui n'ont pas de routes par défaut et qui disposent donc, d'une table de routage complète. Exemples : votre box ne fait pas partie de la DFZ. Les routeurs de bordure de votre FAI font partie de la DFZ. Les routeurs de bordure d'un très gros opérateur comme Level 3 font partie de la DFZ.

Forwarding/Routing En anglais, on appelle le routage effectif des paquets « forwarding » et la construction d'une table de routage « routing ». Tout routeur IP fait du forwarding mais le routage pouvant être statique, tous ne font pas du routing. Ces deux fonctions sont typiquement mises en œuvre par des parties distinctes d'un routeur. Cette distinction explique notamment le nom du paramètre de configuration « ip_forward » sous GNU/Linux : on active le transfert de paquets IP à travers notre machine. D'un côté, on a la RIB (Routing Information Base), la table de routage qui est construite par les protocoles de routage dynamique comme BGP. De l'autre, on a la FIB (Forwarding Information Base) qui contient l'interface de sortie d'un préfixe ainsi que le prochain routeur auquel transmettre directement un paquet destiné à ce préfixe.

Avant d'installer une préfixe dans la FIB, il faut donc calculer l'interface de sortie ainsi que le prochain routeur dans le cas où celui contenu dans la RIB n'est pas un routeur directement accessible.

Internet Routing Registry Un Internet Routing Registry (IRR) est un registre de routage, c'est-à-dire, une base de données publique qui stocke des objets permettant à chaque AS de décrire sa politique de routage dans un format précis qui puisse être parsé par un automate (Routing Policy Specification Language, RPSL). C'est dans cette base de données qu'un AS doit décrire qui sont ses pairs BGP, ce qu'il leur annonce, ce que ses pairs lui annonce, avec quelle priorité il accepte ces annonces, ... Il existe plusieurs IRR, ceux des RIR, à l'exception de l'AfriNIC, bien sûr mais aussi d'autres, celui de Level 3, par exemple. Ces registres, qui permettraient pourtant de créer automatiquement des filtres permettant de rejeter les fausses annonces, sont rarement mis à jour par les opérateurs. Exemple : « aut-num: AS197422 as-name: TETANEUTRAL-NET-AS [...] from AS31576 accept ANY from AS6777 accept ANY from AS6939 accept ANY [...] to AS31576 announce AS197422 to AS6939 announce AS197422 to AS6777 announce AS197422 [...] » Ici, ce FAI toulousain annonce que ses pairs sont GIXE (transitaire), l'AMS-IX (point d'échange néerlandais) et Hurricane Electric (gros opérateur). Tetaneutral.net récupère toutes les routes auprès d'eux sans prioriser un pair vis-à-vis d'un autre. Enfin, Tetaneutral annonce ses préfixes à ses pairs.

Peering Deux AS se sentent mutuellement sur un pied d'égalité (souvent en terme de quantité de trafic) et décident de se retrouver en un même lieu pour interconnecter directement leurs réseaux et s'échanger le trafic destiné à leurs réseaux de manière directe. Cet échange est souvent gratuit mais pas obligatoirement. Les avantages sont financiers, bien sûr, l'échange étant gratuit mais aussi d'ordre technique : on raccourcit drastiquement le chemin entre les deux réseaux (il n'y a plus qu'un seul saut entre les deux), on gagne en indépendance vis-à-vis d'un transitaire (si le transitaire plante, on a encore de la connectivité vers cet AS, et on gagne en redondance : on a une route supplémentaire pour joindre un autre AS. Petite précision : le peering n'est pas transitif ! Si A peere avec B et B peere avec C, A ne peere pas avec C et devra donc avoir un transitaire pour joindre C.

Point d'échange Un point d'échange Internet (Internet eXchange - IX, Internet eXchange Point - IXP, Global Internet eXchange - GIX) est un lieu physique (un datacenter) dans lequel les opérateurs s'interconnectent. Ils facilitent ces interconnexions : sans les points d'échange, chaque opérateur devrait s'interconnecter avec les opérateurs qu'il désire. On aurait donc une explosion combinatoire des possibilités et donc du nombre de câbles nécessaires. L'idée du point d'échange est qu'on ne vient pas s'interconnecter l'un avec l'autre mais on vient tous se connecter au point d'échange. Même si la réalité est plus complexe (services supplémentaires,

problèmes à résoudre, ...), on peut voir un point d'échange comme un simple switch qui relie les opérateurs participants. L'objectif est le même que le peering : relier des opérateurs et réduire ainsi la distance entre eux. Les points d'échange permettent aussi de conserver le trafic destiné à un espace géographique donné dans cet espace (un flux de données Angoulême <-> Bordeaux n'a aucune raison de remonter à Paris). Cela augmente la résilience d'Internet. En France, le trafic est fortement concentré à Paris et il existe peu de points d'échange en région même si l'on peut noter l'existence, entre autres, de l'EuroGIX à Strasbourg, du LyonIX ou du TouIX à Toulouse.

Provider-Aggregatable Un préfixe PA est attribué à un LIR qui le route (un LIR est souvent opérateur) ou le fait router puis le découpe pour ses clients ... ou pas toujours : Opdop alloue des préfixes PA parmi un bloc qu'il ne route pas. Changer de LIR/opérateur signifie aussi changer d'adresses. Les préfixes PA sont apparus avec le découpage classless, afin de limiter le nombre d'entrées dans la table de routage (qui nuit à la performance) engendré par ce découpage : un LIR/opérateur n'annoncera que le préfixe global. Les sous-réseaux resteront un routage interne inconnu du reste d'Internet.

Provider-Independent Un préfixe PI est attribué directement par un RIR à un utilisateur final (même si le RIPE ne fait plus d'allocation directe, c'est-à-dire sans que la demande ne soit formulée par un LIR). Le titulaire du préfixe peut l'annoncer lui-même s'il est opérateur ou le faire annoncer par un opérateur. Il peut donc changer d'opérateur tout en conservant son préfixe. Conséquence logique : un préfixe PI ne peut pas être découpé pour qu'une part soit attribuée à un utilisateur final.

Transit Quand un AS n'a pas une connectivité directe avec un autre AS, il est obligé de recourir aux services d'un transitaire. Un transitaire est un AS qui achemine, à travers son réseau, du trafic qui ne lui est pourtant pas destiné. Un transitaire donne accès à tous les autres réseaux qui composent Internet moyennant rémunération. Ce transitaire peut peerer (donc être présent physiquement dans plusieurs points du globe) et acheter lui-même du transit pour accéder aux destinations où il n'est pas présent mais il doit obligatoirement fournir toutes les routes d'Internet à ses clients. Mais si tous les transitaires fournissent toutes les routes vers toutes les destinations d'Internet, comment en choisir un plutôt qu'un autre ? On tombe là dans des problématiques de qualité des routes proposées. Exemple : vous êtes en France, vous développez un service à destination d'un public purement français et vous avez le choix entre le transitaire A et le transitaire B. A peere avec les réseaux importants français (Orange, Free, ...) à Paris. B peere avec ces mêmes réseaux à Amsterdam. Vous comprenez que, dans votre situation et pour vos besoins, A est mieux que B.

Table des figures

Table des matières