# Enforcing RPKI-Based Routing Policy on the Data Plane at an Internet Exchange

Josh Bailey
Google
joshb@google.com

Dean Pemberton,
Andy Linton
School of Engineering and
Computer Science
Victoria University of
Wellington, New Zealand
{dean,asjl}@ecs.vuw.ac.nz

Cristel Pelsser,
Randy Bush
IIJ
randy@psg.com,
cristel@iij.ad.jp

## ABSTRACT

Over a decade of work has gone into securing the BGP routing control plane. Through all this, there has been an oft-repeated refrain, "It is acknowledged that rigorous control plane verification does not in any way guarantee that packets follow the control plane." We describe what may be the first deployment of data plane enforcement of RPKI-based control plane validation. OpenFlow switches providing an exchange fabric and controlled by a Quagga BGP route server drop traffic for prefixes that have invalid origins without requiring any RPKI support by connected BGP peers. We also report on our operational experience, scaling and administrative problems and, finally provide directions for solving those. The system described is carrying production traffic.

## 1. INTRODUCTION

The Border Gateway Protocol (BGP) [4] is the predominant protocol used on the Internet to exchange reachability information between two different autonomous networks. It is through this successive joining of autonomous networks (known as Autonomous Systems, or ASs) that we arrive at the massive global network we know as the Internet. IP prefixes are associated with ASs that originate these networks in BGP. These advertisements are carried across the Internet and are used by routers (subject to additional operator policy and configuration) to forward packets to their particular destinations. For over a decade, a design group and then the IETF SIDR Working Group have been developing a data repository known as the Resource Public Key Infrastructure (RPKI) [1], and protocols [2] [3] to validate the BGP protocol announcements. In this paper, we rely on the RPKI infrastructure. We use OpenFlow to enforce BGP policies on the data plane, namely here to allow traffic only along routes with a valid origin.

### 1.1 Authenticating routing announcements



While there are mechanisms in place to provide strong authentication between two BGP peers [5], they do not address the validity of the IP prefix advertisements themselves - whether a given AS is entitled to originate a given prefix. Examples of networks being originated by ASs which had no legitimate authority, either by accident or design to do so, have been occuring frequently over a long period of time [6]. More information than is currently available in BGP messages themselves is required to make decisions about the validity of routing advertisements. While operators are able to add configuration (local policy filtering) to address certain cases, like ignoring a remote network's advertisement of networks known to be local, they do not address the more general case. It is also possible to compare routing announcements on an ad hoc basis with databases such as the RIPE IRR, with the caveat that such databases are not intended to be integrated in real time with the control plane [7].

RFC6481 in particular describes how to improve accidental mis-announcements of an IP prefix from the wrong AS. This is referred to as RPKI-based origin validation. Routers use RPKI data to validate the BGP announcements they receive, and drop announcements for prefixes with incorrect origin ASes. It has been suggested however, that even though RPKI-based origin validation looks to cryptographically validate that a given autonomous system is authorised to originate a given prefix, there is no mechanism enforcing this in the 'data plane'.

This is compounded by the fact that, to-date the features required to implement RPKI-based origin validation on commodity networking equipment have been relatively slow to market. There is also some administrative overhead in deploying and operating RPKI (including device certificate management). These factors have had a material impact on the speed at which the Internet at large is able to take advantage of the security enhancements offered by RPKI-based origin validation.

### 1.2 Data plane enforcement via SDN

Software Defined Networking (SDN) may be in a unique position to ease some of these limitations of current systems and enable network operators to enjoy some of the security benefits of RPKI-based origin validation before those features are fully available on their legacy non-SDN capable hardware.

In response to the common security problems associated

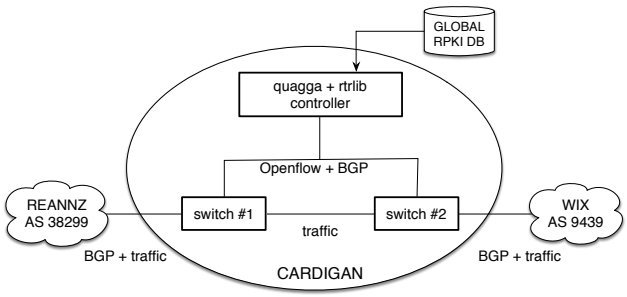


**Figure 1: CARDIGAN Overview**

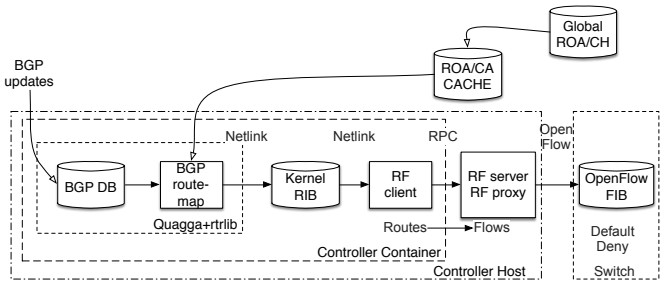


**Figure 2: Translating routes to OpenFlow flows**

with Multilateral Internet Exchanges [8], over the last couple of years [9], the CARDIGAN SDN Internet Exchange (SDX) in New Zealand has been experimenting with the use of OpenFlow controlled exchange switches to enforce Layer-3 BGP policy at Layer-2 in order to prevent providers from accidentally or intentionally using a neighbor as a default exit and similar erroneous forwarding behavior and to enforce BCP38. Exchange peers (REANNZ, and peers on the WIX) use a Quagga-based [10] route server to exchange routes among themselves. Quagga then programs the OpenFlow switches to only forward data for those routes, with all other traffic dropped by default. By incorporating RPKI relying party (RP) functionality into this system we have developed a mechanism with which we can use RPKI derived information to influence whether flows are pushed onto the data path. This way, RPKI policy is enforced across the CARDIGAN exchange without requiring any RPKI capability on the part of the peers. These peers perceive the SDX fabric as a Layer-2 switch plus a route server.

While this prototype has limitations (see following), and vendor support for RPKI does exist in conventional "non-SDN" form [11] the CARDIGAN deployment shows that in principle, any number of BGP peers could connect to such an exchange and have such filtering applied without requiring RPKI support on their own behalf. This enables RPKI relying party functionality be delivered to edge routers that have no native RPKI support. This will help circumvent the scaling and administrative overhead of introducing RPKI validation on all peering routers.

## 2. METHODOLOGY

CARDIGAN is a two-switch, distributed router deployment based on the VANDERVECKEN [12] (which is in turn based on RouteFlow [13]) software stack. Figure 1 shows the two OpenFlow switches which are deployed in different physical locations. Together with the controller, these elements appear to external networks as a single Layer-3 device with one routing table. The switches handle the majority of traffic forwarding in hardware, with certain control plane traffic (including ICMP, ARP, neighbor discovery, and BGP) redirected to the controller, being tunnelled over OpenFlow. The controller calculates the FIB from the RIB and distributes the FIB, in the form of OpenFlow flow rules, to the switches in real time. The controller has configuration associating interconnected ports on the switches, and associating ports on the switches with virtual interfaces on the controller.

The CARDIGAN SDN controller is a Linux (Ubuntu) PC which hosts a container within which Quagga runs. The container provides namespace isolation from the host operating system. This is necessary to keep Quagga's routing table (managing the FIB) separate from the host's (managing the control plane connection to the OpenFlow switches). Since virtual interfaces inside the container are associated with physical interfaces on the switches, processes like Quagga are able to run inside the container and make changes to the RIB without knowing anything about the data plane implementation.

WIX peers and REANNZ routers peer with CARDIGAN's controller using BGP. Previous to this RPKI deployment the BGP protocol stack was provided by the stock Quagga release included with Ubuntu 12.0.4. Now, BGP packets which enter CARDIGAN's data path are delivered to the Quagga process, a normal BGP peering session is established and routes exchanged. Each route to enter the controller's routing table is then translated by RouteFlow into OpenFlow flows. ARP and neighbor discovery are used to resolve nexthops. The flows are then propagated to the switches thus implementing a distributed FIB, which has a default deny policy - flows are proactively programmed as routing announcements are processed, and packets for prefixes not specifically known are dropped.

The API to the distributed FIB is effectively the CARDIGAN controller's kernel routing table running within the container. The interaction between the different elements is shown in Figure 2.

An example of a route to flow transformation is shown in Figures 3, 4 and 5. In this example a route for the 60.234.0.0/16 subnet has been learned with a BGP next hop of 202.7.0.144 (a peer on the WIX exchange), installed in the container's kernel table and translated into two OpenFlow rules.

```
cookie=0x0, duration=623810.66s,
table=0,
n_packets=69883,
n_bytes=6988385,
priority=16560,
ip,in_port=1,
dl_dst=12:a1:a1:a1:a3:02,
nw_dst=60.234.0.0/16
actions=set_field:00:50:56:8a:97:18->eth_src,
set_field:00:12:1e:00:3c:01->eth_dst,
output:2
```

**Figure 4: Example of a flow in the switch facing the WIX**

```
Destination     Gateway         Mask            Flags  MSS Window  irtt Iface
60.234.0.0      202.7.0.144     255.255.0.0     UG     0   0       0    eth4
```

**Figure 3: Example of a route in the kernel routing table of the container**

```
cookie=0x0, duration=618876.366s,
table=0,
n_packets=0,
n_bytes=0,
priority=16560,
ip,
in_port=46,
dl_dst=12:a1:a1:a1:a3:03,
nw_dst=60.234.0.0/16
actions=set_field:12:a1:a1:a1:a3:01->eth_src,
set_field:12:a1:a1:a1:a3:02->eth_dst,
output:48
```

**Figure 5: Example of a flow in the switch facing REANNZ**

Routes such as this one and ARP entries entering and leaving the container's routing and ARP table are noticed via the kernel's netlink interface by RouteFlow's rfclient process (running in the container), which notifies the RFserver and RFproxy components running on the controller host itself. A simple bridge between a dedicated interface within the container and another on the host provides the necessary IP connectivity between container and host.

rfclient translates routes to match/action OpenFlow-style rules, and initiates neighbor discovery to resolve nexthops (by stimulating ARP if necessary). These rules are then sent via RPC to RFserver. RFserver which is running on the host combines this knowledge with configuration of switch ports, and sends the rules to RFproxy. RFproxy then translate the rules into OpenFlow flows, via an OpenFlow connection to the appropriate switch. Routes are withdrawn in a similar way - rfclient commands the removal of rules that match a route to be removed.

The flow in Figure 4 is installed on the switch facing the WIX, as that is where the route's next hop is connected. There is a reciprocal flow, shown in Figure 5, on the other OpenFlow switch so that traffic received from REANNZ can be forwarded to the switch facing the WIX. CARDIGAN uses MAC addresses beginning with octet 12 to denote interfaces that are internal to the distributed fabric, and to enable flow rules to distinguish between hosts on the same local switch and those on a remote switch. These MAC addresses are part of CARDIGAN's configuration.

The flow in Figure 4 has the following match conditions:

- The packet protocol type must be 'IP'
- The ingress port must be port '1'
- The datalink layer (dl) destination address, also known as the destination Ethernet MAC address must be 12:a1:a1:a1:a3:02
- The network layer destination address, also known as the destination IP address must be within the 60.234.0.0/16 subnet

If these are met then the following actions are taken:

- The Ethernet source MAC address is set to 00:50:56:8a:97:18
- The Ethernet destination MAC address is set to 00:12:1e:00:3c:01
- The packet is forwarded out port '2'

Once the flow rules have been installed into the switches, all subsequent packets with destinations matching the subnet 60.234.0.0/16 will be switched in hardware to the nexthop advertised within the peers' BGP session.

There is a 1:1 relationship between OpenFlow flows and the routes with which they are associated. There are also a small number of flows to handle traffic to directly connected hosts and to divert control traffic to the CARDIGAN controller. There are for example specific rules used to direct BGP protocol (TCP port 179) traffic to the controller so that BGP peering sessions can be established, and to allow ARP packets to be passed to the controller so that nexthops can be resolved.

As previously mentioned, a key feature of the route to forwarding table implementation is that the hardware forwarding plane is set to deny all traffic by default. Therefore, because flows are always proactively programmed, if a peer sends traffic to a destination that the controller has not learned via BGP, the packets will be discarded in hardware on the ingress port. This feature alone serves to solve many of the common security problems often found on Internet Exchanges [8].

## 2.1 Implementing RPKI on CARDIGAN

To support RPKI on CARDIGAN two changes were made. The first change was to replace stock Quagga with the rtrlib version [14] which enables Quagga to validate routes via an RPC along with a certificate cache. The RPKI-RP cache, implemented by rcynic, runs on the controller host. The cache is queried by Quagga running inside the container via an RPC, as each route is received from a peer and processed. Being a cache, authenticity or the lack of it may be out of date. This of course is not a limitation of the CARDIGAN implementation in particular.

The following RPKI trust anchors were used in this deployment:

```
ca0.rpki.net
localcert.ripe.net
repo0.rpki.net
repository.lacnic.net
rpki-pilot.lab.dtag.de
rpki-repository.nic.ad.jp
rpki-testbed.apnic.net
rpki.afrinic.net
rpki.apnic.net
rpki.ripe.net
```

ARIN's trust anchor does not appear in the default configuration as it requires the acceptance of additional conditions for use. It was excluded form the deployment for this reason.

```
[Local-Preference Route-Map in place]
route-map rpki permit 10
 match rpki invalid
 set local-preference 10

[Deny Route-Map in place]
route-map rpki deny 10
 match rpki invalid
```

**Figure 6: Views of the Quagga route maps that mark or deny invalid routes**

The second change is to Quagga's configuration, which allows Quagga to discard routes that are specifically invalid (as opposed to being of unknown authenticity).

This was done in two steps. The first step being the route-map in Figure 6 to tag invalid routes with a lower BGP local-preference. This was to understand how many invalid routes were being learned. The second step was to change the route map to the other option shown in Figure 6 to drop any invalid routes entirely.

```
[With Local-Preference Route-Map in place]
#ovs-ofctl dump-flows br0 | grep  103.23.16.0/24

cookie=0x0, duration=620497.082s, table=0,
n_packets=0, n_bytes=0, priority=16640,ip,
in_port=46,dl_dst=12:a1:a1:a1:a3:03,
nw_dst=103.23.16.0/24

actions=
set_field:12:a1:a1:a1:a3:01->eth_src,
set_field:12:a1:a1:a1:a3:02->eth_dst,output:48

[With Deny Route-Map in place]
#ovs-ofctl dump-flows br0 | grep  103.23.16.0/24
#
```

**Figure 7: Dump of the flow rule table on the REANNZ switch showing the invalid route still in place on the switch and then removed**

Invalid routes are tagged with local preference 10, enabling them to be observed in the BGP ("show ip bgp") table as shown in Figure 8. Figure 7 shows that during this phase there is still a corresponding flow-rule for this prefix present on the switches which is not present after the change in Route-Map shown in Figure 6.

## 3. RESULTS

While there is no question that the deployment was limited in size (see Limitations below), a number of practical observations were made from both technical and administrative perspectives.

The CARDIGAN quagga instance has 3 BGP peers. One of these peers is to the REANNZ network and two are to the WIX route servers. The REANNZ peer advertises 5 prefixes while the route servers advertise on the order of 550 routes each (550 and 535 at the time of writing).

In terms of the volume of BGP updates processed from the exchange, from 03/17/14 13:42:40 UTC to 03/24/14 01:00:06 UTC a total of 4173 updates were processed. These were broken down into 2249 BGP advertisements and 1927 BGP withdrawals. A day by day breakdown can be seen in Figure 9.

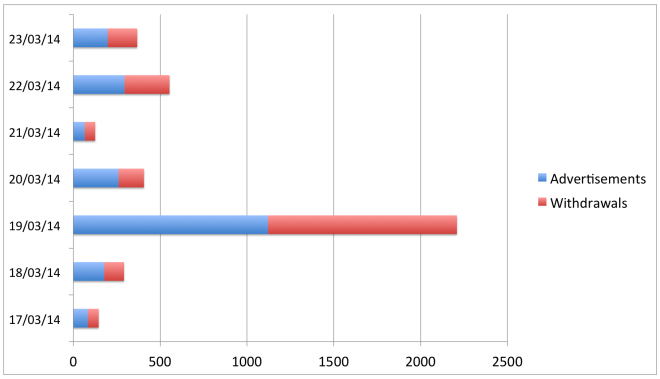


**Figure 9: BGP Updates by CARDIGAN fabric as Advertisements and Withdrawals from 03/17/14 13:42:40 UTC to 03/24/14 01:00:06 UTC**

The number of BGP updates varied widely across the 7 full days being measured. The minimum BGP Advertisements and BGP Withdrawals were 66 and 60 respectively, with the maximum values being 1121 and 1088. The mean of these figures was 314.71 and 271.29 (2dp) with standard deviations of 365.32 and 366.59 (2dp).

A total of 566 routes were learned from the WIX route servers. Of these routes 23 contained a valid ROA while the ROA for 19 of these routes was invalid. The remaining routes had no ROA associated with them.

CARDIGAN successfully prevented these 19 invalid routes from being advertised to REANNZ, and had REANNZ attempted to reach those prefixes the CARDIGAN fabric would have discarded the packets. This was the key goal.

In most of the cases investigated, we found the nineteen invalid routes to be invalid due to the advertised prefix length not matching the ROA. In one case a prefix was being advertised by a party not matching the ROA at all (a legacy deployment). In all cases this represented a connectivity loss to those prefixes from REANNZ, though REANNZ users did not report this loss as a problem.

Operationally it was possible to use OpenFlow flow counters, to measure traffic to each prefix. We observed that switch1 received 18.81 GB of traffic from REANNZ while switch2 received 20.18 GB from the WIX from March 17, 2014 to March 24, 2014. Figure 10 shows the percentage of this traffic per flow rule (essentially an IP destination prefix). In our case, a single entry is used to forward most of the traffic. This is true for both switches: 89.31% of the traffic for the switch facing REANNZ and 83.22% for the switch facing WIX. Switch1 only drops 959 bytes while switch2 drops 42.60 KB of traffic over a period of a week. These counters made analysis of the consequences of route filtering very easy - it could make an informed prediction about the impact of dropping a flow for an invalid prefix based on historical traffic.

There were a number of administrative issues noted. For example, it was observed that APNIC does not allow ROAs to be asserted within their trust anchor by organisations without full APNIC member status. This has the effect of

```
I*  103.23.16.0/24    202.7.0.222                10       0 9439 132040 i
I*>                   202.7.0.221                10       0 9439 132040 i
I*  103.23.17.0/24    202.7.0.222                10       0 9439 132040 i
I*>                   202.7.0.221                10       0 9439 132040 i
```

**Figure 8: Output of the "show ip bgp" command. Invalid routes are tagged with a local preference 10.**

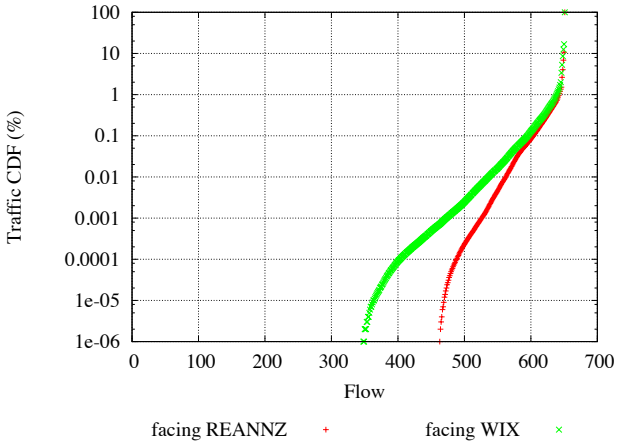


**Figure 10: Normalized cumulative ditribution of the traffic on the two switches**

excluding a number of Asia Pacific based organisations with legacy or historical addresses who are APNIC members of lesser classes. This particular policy is not uniform across all the RIRs. This question is under current discussion.

## 4. LIMITATIONS

There are a number of limitations in the CARDIGAN testbed which are primarily specific to the CARDIGAN switch hardware itself - Pica8 3290 and 3780 switches - and to the experimental nature of the entire software stack.

At present these switches allow only TCAM (as opposed to FIB) resources to be programmed with OpenFlow, which puts a limit (1k to 2k) flows total. The current Pica8 software (2.1) also does not support IPv6, though the upcoming 2.2 software does.

To address this flow scaling limitation FIB compression techniques like route summarization (maintaining correct forwarding behavior while reducing the number of rules, as demonstrated in a separate experiment by the FRONT LINE ASSEMBLY project [15]) could be used. A 40% compression of the FIB was reached between ESnet and REANNZ, two New Zealand national internet service providers.

Future generations of OpenFlow hardware support more flows and multiple tables - NoviFlow's NPU-based switches are an example. Also, as OpenFlow implementations mature on existing hardware it is expected non-TCAM resources will be available for flow matching. An example is host MAC address matching hardware which is commonly used to match a given destination MAC address specifically. While OpenFlow 1.3 was used in this deployment, the switches provided only a single table. This precluded experimentation with structuring flow tables as trees or pipelines rather than a single TCAM-based table.

The prototype's RIB to FIB convergence time is slow - measured at 10 to 20ms per flow added or removed. There are a number of CARDIGAN implementation specific reasons for this - including routes passing through multiple python processes with extensive string handling. The flow modification performance of the Pica8 switches also has not been measured independently. A faster implementation would likely not involve python.

Today, control traffic is forwarded to the control container over an OpenFlow connection. I.e. ARP, BGP etc. must be passed from the data plane to the switch's CPU, encapsulated in OpenFlow, sent to the controller, and then decapsulated. It would be better to dedicate ports in the data plane and a controller port to control traffic, which would avoid the switch CPU and encapsulation.

Policy (specifically, what to do with invalid routes) is fairly coarse grained via the route-map mechanism - while it is possible to change/suppress advertisements, more flexible mechanisms would be useful for wider/larger deployments. For example, in the case of a catastrophic RIR failure or compromise, large numbers of advertisements could be affected or dropped. One possible solution might be to prompt for operator intervention if an automated policy change would cause a very large traffic change.

## 5. RELATED WORK

The route server in [16] is coupled with an OpenFlow controller that installs flows for the BGP routes into the switches owned by the IX. [16] is the ancestor of this work. In this work, only routes with a valid origin AS are installed in the switches. RPKI-based routing policies are enforced. SDX [17] also proposes a controller for Internet Exchanges. Its focus is different in that ISPs are given the possibility to configure the exchange to implement policies on their behalf.

B4 [18] and SWAN [19] are other examples of SDN deployments in production networks. These are private WANs connecting Google's and Microsoft's data centers, respectively. The traffic demand is mostly known a priori, making it suitable for a controller to pre-configure various number of paths between pairs of DCs depending on the expected load.

Not many techniques are currently available to enforce congruence between the data and routing planes. Among these, RPF-check (Reverse Path Forwarding check) [20] allow verification that packets with a given IP source can be expected on a particular interface or at the router by checking if there is a route for the source in the table. This mechanism provides some protection against IP spoofing. Here, the protection provided differs. First, packets to destinations not advertised by the IX member are not forwarded to the members' infrastructure. This is a destination-based congruence check. Second, BGP routes of invalid origin are discarded.

## 6. CONCLUSION

Enforcing control plane policy consistently and scalably

in the data plane, whether at the routing level or the MAC level has been challenging. It is possible to address enforcement and authenticity problems in and at different layers (for example, using MACSEC to help authenticate a peer). However until the advent of OpenFlow in particular the data plane was not generically nor directly programmable - this programmability represents an opportunity to apply policy directly to the data plane.

As discussed in the limitations section, scalability challenges remain - OpenFlow implementations are still relatively primitive and incomplete, and lower end hardware does not provide the necessary flow matching resources for more than a few thousand proactively programmed flows. However higher end OpenFlow hardware (Eg, NoviFlow 200) supports many hundreds of thousands of flows and it can be imagined more hardware with even more capabilities will become available in the future.

This deployment also suggests a method to support RPKI in an environment with non-SDN peers - the peers do not need to support RPKI to gain the benefits of the exchange fabric's capabilities. Further exploration and leverage of this proxy programmability/capability will continue on the CARDIGAN platform.

## Acknowledgment

The authors would like to thank Stephen Stuart for his insight on the project.

## 7. REFERENCES

[1] G. Huston, R. Loomans, and G. Michaelson, "A Profile for Resource Certificate Repository Structure," RFC 6481 (Proposed Standard), Internet Engineering Task Force, Feb. 2012.

[2] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol," RFC 6810 (Proposed Standard), Internet Engineering Task Force, Jan. 2013.

[3] P. Mohapatra, J. Scudder, D. Ward *et al.*, "BGP Prefix Origin Validation," RFC 6811 (Proposed Standard), Internet Engineering Task Force, Jan. 2013.

[4] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task Force, Jan. 2006, updated by RFCs 6286, 6608, 6793.

[5] A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option," RFC 2385 (Proposed Standard), Internet Engineering Task Force, Aug. 1998, obsoleted by RFC 5925, updated by RFC 6691.

[6] (2008, March) Youtube hijacking: A RIPE NCC RIS case study. [Online]. Available: http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

[7] (2014) RIR statistics exchange format. [Online]. Available: https://www.apnic.net/publications/media-library/documents/resource-guidelines/rir-statistics-exchange-format

[8] M. Jager. (2012) Securing IXP connectivity.

[9] D. Pemberton. NZ scores first OpenFlow controlled connection to an IX. [Online]. Available: http://list.waikato.ac.nz/pipermail/nznog/2012-December/019635.html

[10] Quagga project. [Online]. Available: http://www.nongnu.org/quagga/

[11] (2011, December) RPKI: Router Configuration. [Online]. Available: http://www.ripe.net/lir-services/resource-management/certification/router-configuration

[12] J. Bailey. (2012, October) VANDERVECKEN: a reference OpenFlow-controlled router/MPLS implementation for network research. [Online]. Available: http://goo.gl/2eQskD

[13] Routeflow project. [Online]. Available: https://sites.google.com/site/routeflow/

[14] GitHub: Quagga with RPKI-RTR prefix origin validation support. [Online]. Available: https://github.com/rtrlib/quagga-rtrlib

[15] J. Bailey, S. Whyte, D. Hall *et al.*, "Front-Line Assembly: First international BGP peering using SDN in production between two national-scale network providers," Demo at Open Networking Summit 2013 (ONS 2013), April 2013. [Online]. Available: http://homepages.ecs.vuw.ac.nz/foswiki/pub/Users/Josh/TREEHOUSE/fla-poster.pdf

[16] J. P. Stringer, Q. Fu, C. Lorier *et al.*, "Cardigan: Deploying a distributed routing fabric," in *HotSDN 2013 (Poster session)*, August 2013.

[17] A. Gupta, M. Shahbaz, L. Vanbever *et al.*, "SDX: A Software Defined Internet Exchange," Georgia Institute of Technology, Tech. Rep. GT-CS-13-06, November 2013.

[18] S. Jain, A. Kumar, S. Mandal *et al.*, "B4: Experience with a globally-deployed Software Defined WAN," in *SIGCOMM*, 2013.

[19] C.-Y. Hong, S. Kandula, R. Mahajan *et al.*, "Achieving high utilization with software-driven WAN," in *SIGCOMM*, 2013.

[20] F. Baker and P. Savola, "Ingress Filtering for Multihomed Networks," RFC 3704 (Best Current Practice), Internet Engineering Task Force, March 2004.