NIST SPECIAL PUBLICATION 1800-14C

Protecting the Integrity of Internet Routing:

Border Gateway Protocol (BGP) Route Origin Validation

Volume C:
How-To Guides

William Haag
Applied Cybersecurity Division
Information Technology Laboratory

Doug Montgomery
Advanced Networks Technology Division
Information Technology Laboratory

Allen Tan
The MITRE Corporation
McLean, VA

William C. Barker
Dakota Consulting
Silver Spring, MD

August 2018

DRAFT

This publication is available free of charge from:
https://www.nccoe.nist.gov/projects/building-blocks/secure-inter-domain-routing

National Institute of Standards and Technology
U.S. Department of Commerce

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST or NCCoE, nor is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1800-14C, Natl. Inst. Stand. Technol. Spec. Publ. 1800-14C, 61 pages, (August 2018), CODEN: NSPUE2

FEEDBACK

You can improve this guide by contributing feedback. As you review and adopt this solution for your own organization, we ask you and your colleagues to share your experience and advice with us.

Comments on this publication may be submitted to: sidr-nccoe@nist.gov.

Public comment period: August 30, 2018 through October 15, 2018

All comments are subject to release under the Freedom of Information Act (FOIA).

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE), a part of the National Institute of Standards and Technology (NIST), is a collaborative hub where industry organizations, government agencies, and academic institutions work together to address businesses' most pressing cybersecurity issues. This public-private partnership enables the creation of practical cybersecurity solutions for specific industries, as well as for broad, cross-sector technology challenges. Through consortia under Cooperative Research and Development Agreements (CRADAs), including technology partners—from Fortune 50 market leaders to smaller companies specializing in IT security—the NCCoE applies standards and best practices to develop modular, easily adaptable example cybersecurity solutions using commercially available technology. The NCCoE documents these example solutions in the NIST Special Publication 1800 series, which maps capabilities to the NIST Cybersecurity Framework and details the steps needed for another entity to recreate the example solution. The NCCoE was established in 2012 by NIST in partnership with the State of Maryland and Montgomery County, Md.

To learn more about the NCCoE, visit https://www.nccoe.nist.gov/. To learn more about NIST, visit https://www.nist.gov.

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication Series 1800) target specific cybersecurity challenges in the public and private sectors. They are practical, user-friendly guides that facilitate the adoption of standards-based approaches to cybersecurity. They show members of the information security community how to implement example solutions that help them align more easily with relevant standards and best practices, and provide users with the materials lists, configuration files, and other information they need to implement a similar approach.

The documents in this series describe example implementations of cybersecurity practices that businesses and other organizations may voluntarily adopt. These documents do not describe regulations or mandatory practices, nor do they carry statutory authority.

ABSTRACT

The Border Gateway Protocol (BGP) is the default routing protocol to route traffic among internet domains. While BGP performs adequately in identifying viable paths that reflect local routing policies and preferences to destinations, the lack of built-in security allows the protocol to be exploited by route hijacking. Route hijacking occurs when an entity accidentally or maliciously alters an intended route. Such attacks can (1) deny access to internet services, (2) detour internet traffic to permit eavesdropping and to facilitate on-path attacks on end points (sites), (3) misdeliver internet network traffic to malicious end points, (4) undermine internet protocol (IP) address-based reputation and filtering systems, and (5) cause routing instability in the internet. This document describes a security platform that

demonstrates how to improve the security of inter-domain routing traffic exchange. The platform provides route origin validation (ROV) by using the Resource Public Key Infrastructure (RPKI) in a manner that mitigates some misconfigurations and malicious attacks associated with route hijacking. The example solutions and architectures presented here are based upon standards-based, open-source, and commercially available products.

KEYWORDS

AS, autonomous systems, BGP, Border Gateway Protocol, DDoS, denial-of-service (DoS) attacks, internet service provider, ISP, Regional Internet Registry, Resource Public Key Infrastructure, RIR, ROA, route hijack, route origin authorization, route origin validation, routing domain, ROV, RPKI

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

Name	Organization
Tim Battles	AT&T
Jay Borkenhagen	AT&T
Chris Boyer	AT&T
Nimrod Levy	AT&T
Kathryn Condello	CenturyLink
Christopher Garner	CenturyLink
Peter Romness	Cisco Systems
Tony Tauber	Comcast
Jonathan Morgan	Juniper Networks
Carter Wyant	Juniper Networks
Oliver Borchert	NIST ITL Advanced Networks Technologies Division

Name	Organization
Katikalapudi Sriram	NIST ITL Advanced Networks Technologies Division
Sean Morgan	Palo Alto Networks
Tom Van Meter	Palo Alto Networks
Andrew Gallo	The George Washington University
Sophia Applebaum	The MITRE Corporation
Yemi Fashina	The MITRE Corporation
Susan Prince	The MITRE Corporation
Susan Symington	The MITRE Corporation

○

The Technology Partners/Collaborators who participated in this build submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build this example solution. We worked with:

Technology Partner/Collaborator	Build Involvement
AT&T	Subject Matter Expertise
CenturyLink	1 gigabit per second (Gbps) Ethernet Link Subject Matter Expertise
Cisco	7206 VXR Router v15.2 ISR 4331 Router v16.3 2921 Router v15.2 IOS XRv 9000 Router v6.4.1 Subject Matter Expertise
Comcast	Subject Matter Expertise

Technology Partner/Collaborator	Build Involvement
Juniper Networks	MX80 3D Universal Edge Router v15.1R6.7 Subject Matter Expertise
Palo Alto Networks	Palo Alto Networks Next-Generation Firewall PA-5060 v7.1.10 Subject Matter Expertise
The George Washington University	Subject Matter Expertise

Contents

List of Figures

1 Introduction

The following guides show information technology (IT) professionals and security engineers how we implemented the example Secure Inter-Domain Routing (SIDR) Project solution for Resource Public Key Infrastructure (RPKI)-based route origin validation (ROV). We cover all of the products employed in this reference design. We do not recreate the product manufacturers' documentation, which is presumed to be widely available. Rather, these guides show how we incorporated the products together in our environment.

Note: These are not comprehensive tutorials. There are many possible service and security configurations for these products that are out of scope for this reference design.

1.1 Practice Guide Structure

This National Institute of Standards and Technology (NIST) Cybersecurity Practice Guide demonstrates a standards-based reference design and provides users with the information they need to replicate the SIDR RPKI-based ROV solution. This reference design is modular and can be deployed in whole or in parts.

NIST Special Publication (SP) 1800-14 contains three volumes:

- NIST SP 1800-14A: *Executive Summary*
- NIST SP 1800-14B: *Approach, Architecture, and Security Characteristics* – what we built and why
- NIST SP 1800-14C: *How-To Guides* – instructions for building the example solution **(you are here)**

Depending on your role in your organization, you might use this guide in different ways:

Business decision makers, including chief security and technology officers, will be interested in the *Executive Summary* (NIST SP 1800-14A), which describes:

- The challenges that enterprises face in implementing and maintaining route origin validation
- An example solution built at the National Cybersecurity Center of Excellence (NCCoE)
- Benefits of adopting the example solution

Technology or security program managers who are concerned with how to identify, understand, assess, and mitigate risk will be interested in NIST SP 1800-14B, which describes what we did and why. The following sections will be of particular interest:

- Section 4.4.3, Risks, provides a description of the risk analysis we performed
- Section 4.4.4, Cybersecurity Framework Functions, Categories, and Subcategories Addressed by the Secure Inter-Domain Routing Project, maps the security characteristics of this example solution to cybersecurity standards and best practices

If you are a technology or security program manager, you might share the *Executive Summary,* NIST SP 1800-14A, with your leadership team members to help them understand the importance of adopting the standards-based SIDR RPKI-based ROV solution.

IT professionals who want to implement an approach like this can use the How-To portion of the guide, NIST SP 1800-14C, to replicate all or parts of the build created in our lab. The How-To guide provides specific product installation, configuration, and integration instructions for implementing the example solution. We do not recreate the product manufacturers' documentation, which is generally widely available. Rather, we show how we incorporated the products together in our environment to create an example solution.

This guide assumes that IT professionals have experience implementing security products within the enterprise. While we have used a suite of commercial products to address this challenge, it is not NIST policy to endorse any particular products. Your organization can adopt this solution or one that adheres to these guidelines in whole, or you can use this guide as a starting point for tailoring and implementing parts of an RPKI-based ROV solution. Your organization's security experts should identify the products that will best integrate with your existing tools and IT system infrastructure. We hope that you will seek products that are congruent with applicable standards and best practices. Section 4.5, Technologies, of NIST SP 1800-14B lists the products that we used and maps them to the cybersecurity controls provided by this reference solution.

A NIST Cybersecurity Practice Guide does not describe "the" solution, but a possible solution. This is a draft guide. We seek feedback on its contents and welcome your input. Comments, suggestions, and success stories will improve subsequent versions of this guide. Please contribute your thoughts to sidr-nccoe@nist.gov.

1.2 Build Overview

This NIST Cybersecurity Practice Guide addresses the challenge of using existing protocols to improve the security of inter-domain routing traffic exchange in a manner that mitigates accidental and malicious attacks associated with route hijacking. It implements and follows various Internet Engineering Task Force (IETF) Request for Comments (RFC) documents that define RPKI-based Border Gateway Protocol (BGP) ROV, such as RFC 6480, RFC 6482, RFC 6811, and RFC 7115, as well as recommendations of NIST

SP 800-54, *Border Gateway Protocol Security*. To the extent practicable from a system composition point of view, the security platform design, build, and test processes have followed NIST SP 800-160, *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*.

The ROV capabilities demonstrated by the proof-of-concept implementation described in this Practice Guide improve inter-domain routing security by using standards-conformant security protocols to enable an entity that receives a route advertisement to validate whether the autonomous system (AS) that has originated it is in fact authorized to do so.

In the NCCoE lab, the team built an environment that resembles portions of the internet. The SIDR lab architecture is depicted in Figure 1-1 and Figure 1-2. It consists of virtual and physical hardware, physical links to ISPs, and access to the Regional Internet Registries (RIRs). The physical hardware mainly consists of the routers performing ROV, workstations providing validator capabilities, and firewalls that protect the lab infrastructure. The virtual environment hosts the RPKI repositories, validators, and caches used for both the hosted and delegated RPKI scenarios. The architecture is organized into separate virtual local area networks (VLANs), each of which is designed to represent a different AS. For example, VLAN 1 represents an ISP with AS 64501, VLAN 2 represents the enterprise network of an organization with AS 64502, and VLAN 3 represents an ISP with AS 64503.

The configurations in this document provide a baseline for completing all the test cases that were performed for the project.

There are two environments that are used: test harness and live data.

- The test harness environment consists of physical/virtual routers, a lab RPKI repository, RPKI validators, and simulation tools (or test harness). The physical and virtual routers in this environment are from Cisco and Juniper. The lab RPKI repository is configured using the RPKI.net tool. The RPKI caches in this environment are the Réseaux IP Européens Network Coordination Centre (RIPE NCC) validator and the RPKI.net validator. The test harness simulates BGP routers sending and receiving advertisements and emulates RPKI data being sent from validators/caches. There are two components of the test harness: the BGPSEC-IO (BIO) traffic generator and collector, which produces BGP routing data, and the SRx-RPKI validator cache test harness, which simulates RPKI caches.
- The live data environment leverages many of the same components from the test harness environment. The difference is that this environment leverages live data from the internet, rather than uses emulated BGP advertisements and RPKI data. The physical and virtual routers in this environment are from Cisco and Juniper. The lab RPKI repository is configured using the RPKI.net tool. Repositories from the RIRs (American Registry for Internet Numbers [ARIN], RIPE NCC, African Network Information Center [AFRINIC], Latin America and Caribbean Network Information Center [LACNIC], and Asia-Pacific Network Information Center [APNIC]) are also used to receive real-world route origin authorization (ROA) data. The RPKI caches in this

environment are the RIPE NCC validator and the RPKI.net validator. A physical wide area network (WAN) link is used to connect to CenturyLink to receive a full BGP table and to connect to the RIRs.

Figure 1-1 Test Harness Environment for SIDR RPKI-Based ROV Solution Testing

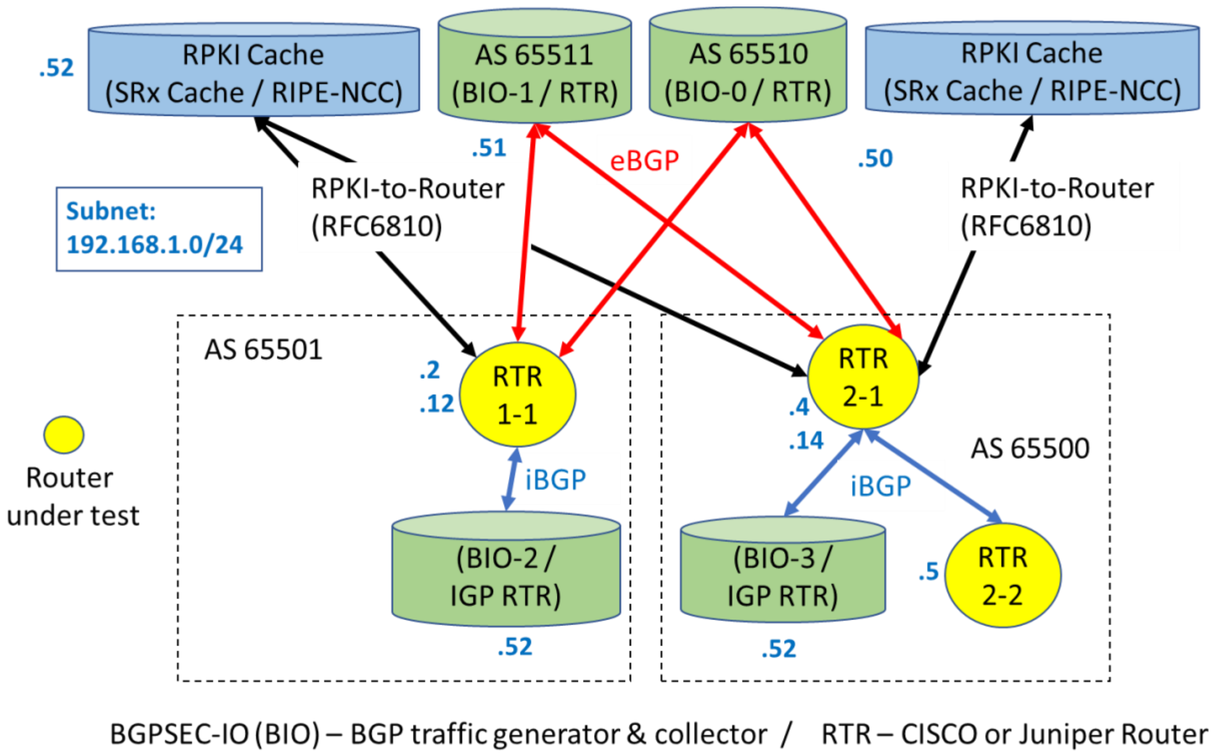
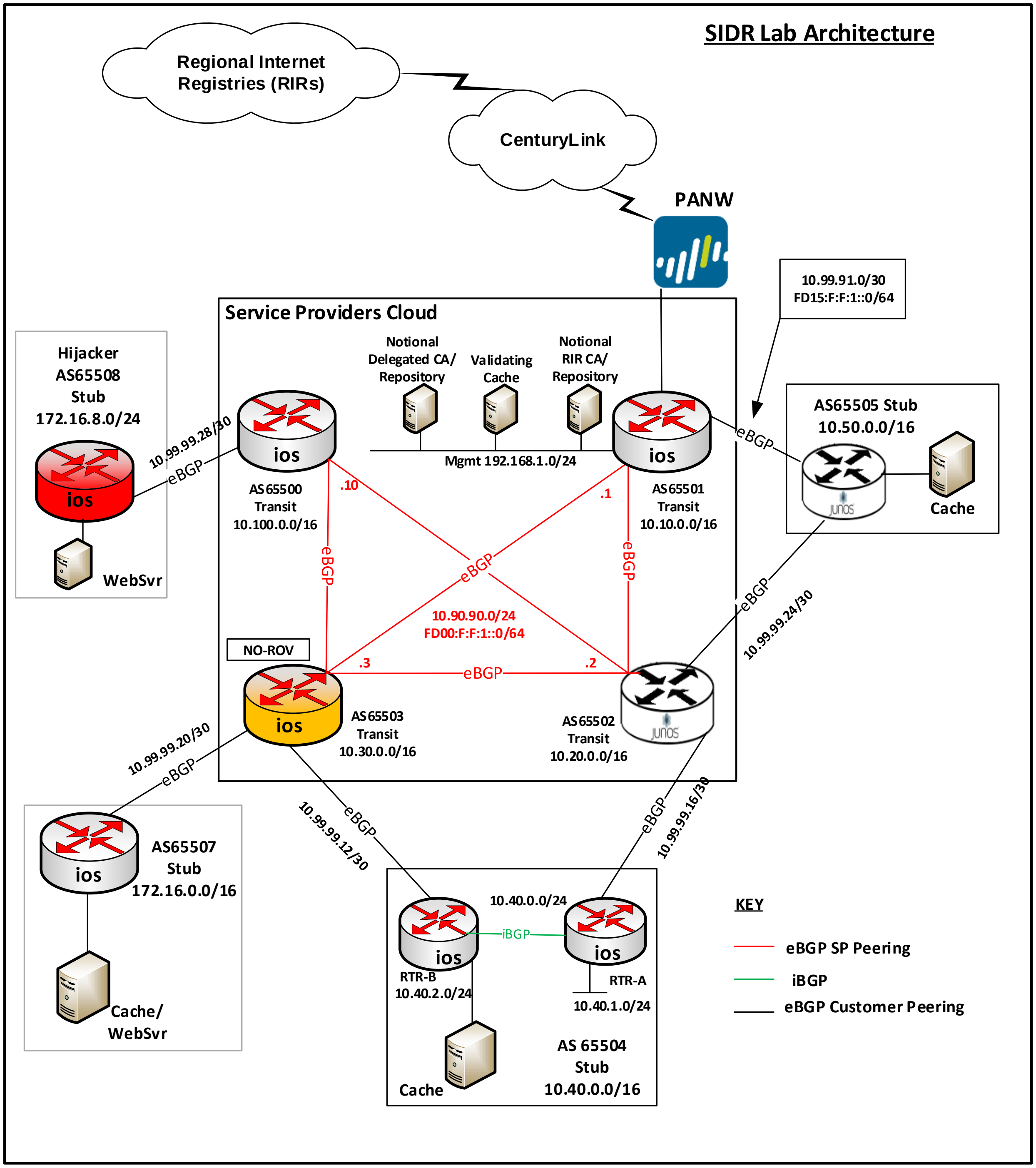


Figure 1-2 Live Data Environment for SIDR RPKI-Based ROV Solution Testing



1.3 Typographic Conventions

The following table presents typographic conventions used in this volume.

Typeface/Symbol	Meaning	Example
Italics	filenames and pathnames references to documents that are not hyperlinks, new terms, and placeholders	For detailed definitions of terms, see the *CSRC.NIST.GOV Glossary.*
Bold	names of menus, options, command buttons, and fields	Choose **File > Edit.**
`Monospace`	command-line input, on-screen computer output, sample code examples, status codes	`Mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's National Cybersecurity Center of Excellence are available at http://www.nccoe.nist.gov

2 Product Installation Guides

This section of the Practice Guide contains detailed instructions for installing and configuring all of the products used to build an instance of the SIDR RPKI-based ROV example solution. The main components of the lab build consist of ROV-enabled routers, RPKI repositories, RPKI validators / validating caches (VCs), a live internet circuit, and firewalls.

2.1 RPKI Validators

The RPKI validator receives and validates ROAs from the RPKI repositories of the trust anchors and delegated repositories. Currently, there are five trust anchors, all of which are managed by the RIRs: AFRINIC, APNIC, ARIN, LACNIC, and the RIPE NCC. A subset of the data from ROAs, called validated ROA payload (VRP), is then retrieved from the local RPKI validator by an RPKI-capable router to perform ROV of BGP routes.

In this lab build, two RPKI validators (also referred to as VCs) are tested: the RIPE NCC RPKI validator and the Dragon Research RPKI.net validator.

2.1.1 RIPE NCC RPKI Validator Configuration/Installation

The RIPE NCC RPKI validator is developed and maintained by RIPE NCC [RIPE Tools]. This validator tool is free and open-source. The version used in the build is 2.24. It is available for download at https://www.ripe.net/manage-ips-and-asns/resource-management/certification/tools-and-resources.

System requirements: a UNIX-like operating system (OS), Java 7 or 8, rsync, and 2 gigabytes (GB) of free memory.

Lab setup: CentOS 7 minimal install, Java 8, rsync, one central processing unit (CPU), 6 GB memory, and running on a virtual machine (VM) on VMware ESXi.

For release notes, installation information, and source code, please view https://github.com/RIPE-NCC/rpki-validator/blob/master/rpki-validator-app/README.txt.

1. Use the CentOS template to create the VM with the system requirements provided above.
    a. Put the VM in the proper VLAN.
2. Install Java (must be Oracle 8) and open firewall to allow rsync.
3. In the VM, create a folder under home called "RPKI".
    a. **`# mkdir RPKI`**
    b. **`# cd RPKI`**
4. Download and install the RIPE NCC RPKI validator software in the VM.
    a. **`# tar -xvf rpki-validator-app-2.24-dist.tar.gz`**
5. Set *JAVA_HOME* (only if the application complains that it does not see the *JAVA_HOME* path).
    a. **`# cd /etc/environment`**
        i. **`# nano environment`**

ii. `# JAVA_HOME="/usr"`

b. Source it and check echo.

i. `# source /etc/environment`

ii. `# Echo $JAVA_HOME`

6. Reboot the server.
7. Start the RPKI cache.

a. `# ./rpki-validator.sh start`

8. Using a web browser, connect to the validator software that you just installed, by typing http://ip-address:8080 into the browser search window, replacing "ip-address" with the internet protocol (IP) address of the VM that you just created in step 1. (i.e., http://192.168.1.124:8080).
9. Once the validator is up, it receives data from the following RIR repositories: AFRINIC, APNIC, LACNIC, and RIPE NCC.

a. To retrieve ROAs from the ARIN repository, download the Trust Anchor Locator (TAL) file from https://www.arin.net/resources/rpki/tal.html.

b. Stop the validator.

i. `# ./rpki-validator.sh stop`

c. Put the file in the *TAL* sub-directory.

d. Restart the validator.

i. `# ./rpki-validator.sh start`

2.1.2 Dragon Research RPKI.net Validator Configuration/Installation

The Dragon Research Labs-developed RPKI.net toolkit contains both a VC and a certificate authority (CA). This section discusses the VC only.

System requirements: Ubuntu 16.04 Xenial server, 32 GB of hard disk, 1 GB of random access memory (RAM), and a minimum of one CPU.

Lab setup: Ubuntu 16.04 Xenial server, rsync, one CPU, 6 GB memory, and running on a VM on VMware ESXi.

For release notes, installation information, and additional information, please view https://github.com/dragonresearch/rpki.net/blob/master/doc/quickstart/xenial-rp.md.

```
# wget -q -O
/etc/apt/sources.list.d/rpki.list https://download.rpki.net/APTng/rpki.xenial.l
ist
```

You may get a message that says that there were errors (i.e., “the following signatures couldn’t be verified because the public key is not available”). To fix this, use the following command, along with the key that showed up on the error:

```
# apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 40976EAF437D05B5
```

Note: *40976EAF437D05B5* is an example. Use the exact key that showed up in the error.

Reference: https://chrisjean.com/fix-apt-get-update-the-following-signatures-couldnt-be-verified-because-the-public-key-is-not-available/.

```
# apt update
```

```
# apt install rpki-rp
```

This should install the VC. Next, access the VC by opening a browser and typing http://192.168.2.106/rcynic into the search window.

Note: It takes up to an hour to completely update. The proper Uniform Resource Locator (URL) will not show up until then. Just wait for it. You will see a parent folder directory in the URL during that time. Once it’s ready, charts about the repositories from the different RIRs will show up.

Check to see if the VC is running by entering the following command:

```
# ps -aux | grep rpki
```

2.2 RPKI CA and Repository

The delegated model of RPKI for ROA creation and storage requires that two components be set up, operated, and maintained by the address holder: a CA and a repository. Currently, only the Dragon Research RPKI.net toolkit provides the components needed to set up a delegated model.

2.2.1 Dragon Research RPKI.net CA and Repository Configuration/Installation

The setup for the CA and repository is different from the setup for the relying-party VC.

System requirements: Ubuntu 16.04 Xenial server, 32 GB of hard disk, 1 GB of RAM, and a minimum of one CPU.

Lab setup: Ubuntu 16.04 Xenial server, rsync, one CPU, 6 GB memory, and running on a VM on VMware ESXi.

For release notes, installation information, and additional information, please view https://github.com/dragonresearch/rpki.net/blob/master/doc/quickstart/xenial-ca.md.

Steps for installing the rpki-ca (the CA software) toolkit for this lab build were different from the instructions provided by the GitHub documentation. Guidance for the lab build is provided below.

2.2.1.1 Assumptions

Prior to installing rpki-ca and rpki-rp (the repository software), ensure that you are working with two hosts running the Ubuntu Xenial server. In our setup, we will call one host *primary_root* (parent) and the other host *remote_child* (child); both are running the Ubuntu Xenial server.

2.2.1.2 Installation Instructions

Run the initial setup to install rpki-ca. Follow the steps in the Xenial guide up to "CA Data initialization".

Execute the steps under rcynic and rsyncd, specifically the "cat" commands that are listed.

2.2.1.3 Getting rcynic to Run

1. It's important to note that the rcynic software will NOT be installed correctly. You will need to add the following line to */var/spool/cron/crontabs/rcynic*:

   ```
   */10 * * * * exec /usr/bin/rcynic-cron
   ```

   a. This ensures that the rcynic software will be run periodically to update the certificates. This should be done on both hosts. Rcynic is designed to run periodically by default.

   b. Rcynic will error out when external TAL files are called. Delete all repository files in the trust-anchors folder. To do this, run the following command:

      ```
      # rm /etc/rpki/trust-anchors/*
      ```

      i. This should be done on both hosts.

2. The next step is to edit the */etc/rpki.conf* file.

   a. On the host that we will be calling *primary_root*, make the following changes:

      i. Change the handle to *primary_root*.

      ii. Change rpkic_server_host to *0.0.0.0*.

      iii. Change irdb_server_host to *0.0.0.0*.

      iv. Set run_pubd to *yes*.

      v. Change pubd_server_host to *0.0.0.0*.

      This should be sufficient for the changes on primary_root.

b. On the host that we will be calling *remote_child*, make the following changes to */etc/rpki.conf*:

   i. Change the handle to *remote_child*.

   ii. Change rpkic_server_host to *localhost*.

   iii. Change irdb_server_host to *localhost*.

   iv. Set run_pubd to *no*.

   v. Change pubd_server_host to *primary_root*.

   This last change means that remote_child will look to primary_root as the publication server rather than running its own. To access primary_root, remote_child will need a Domain Name System entry for primary_root.

   1) To create this, first find primary_root's IP address by running **`ifconfig`** on primary_root. In our setup, this IP address is 192.168.2.115.

   2) Then, on remote_child, we add the following line to the */etc/hosts* file:

```
192.168.2.115: primary_root :(Replacing the IP address with
whatever IP address is currently assigned to primary_root.)
```

At this point, rcynic, rpkic, and rsyncd should all be set up.

3. On both hosts, run the following commands to reboot the services:

```
# systemctl restart xinetd
```

```
# systemctl restart rpki-ca
```

2.2.1.4 GUI Setup

1. Set up the graphical user interface (GUI) on both VMs by running the following command:

```
# rpki-manage createsuperuser
```

2. Fill in the details appropriately. Verify that each GUI is up by opening a browser and visiting https://127.0.0.1 on both hosts.

2.2.1.5 Root CA Repository Setup

1. For simplicity, create a folder named */root/CA-stuff* on both VMs. Change the directory into this folder for both VMs.

2. Now, we will set up primary_root as a root server for all resources.

a. On primary_root, run the following command:

```
# rpkic create_identity primary_root
```

This will produce a file named *primary_root.identity.xml.*

b. Next, run the following command:

```
# rpkic configure_root
```

This will produce a file named *primary_root.primary_root.repository-request.xml*. We will return to this file later.

c. Now, run the following command:

```
# rpkic -i primary_root extract_root_certificate
# rpkic -i primary_root extract_root_tal
```

These commands will respectively produce a *.cer* file and a *.tal* file.

d. Copy both of these files into the */usr/share/rpkic/rrdp-publication* folder. (Note: This step may not be necessary.)

e. Copy the *.tal* file to */etc/rpki/trust-anchors.* This step configures rcynic to look at this node as a repository.

f. Now, we will copy the *.tal* file from primary_root to remote_child. One way to do this is with rsync as follows:

   i. Copy the *.tal* file to */usr/share/rpki/publication* on primary_root.

   ii. On remote_child, run the following command to verify that rsync is working, replacing the IP address as appropriate in the command below:

   ```
   # rsync rsync://192.168.2.115/rpki
   ```

   iii. If the above runs correctly, copy the *.tal* file, replacing <file> as appropriate in the command below:

   ```
   # rsync rsync://192.168.2.115/rpki/<file>.tal /etc/rpki/trust-anchors
   ```

Now, primary_root's *.tal* file should be on both VMs in the */etc/rpki/trust-anchors* directory.

g. We now want to update rcynic. To force it to synchronize, we run the following command on both VMs:

```
# sudo -u rpki python /usr/bin/rcynic-cron
```

i. To verify that rcynic works, visit https://127.0.0.1/rcynic on both VMs.

h. We return to setting up primary_root.

i. On primary_root, find the file named *primary_root.primary_root.repository-request.xml.* Once in the right directory, run the following command:

```
# rpkic configure_publication_client
primary_root.primary_root.repository-request.xml
```

This should produce a file named *primary_root.repository-response*.

ii. With this file, run the following command:

```
# rpkic configure_repository primary_root.repository-response
```

Now, primary_root should be set up.

i. On primary_root, visit https://127.0.0.1 and log in. You should see primary_root as a repository at the bottom of the page.

2.2.1.6 Child CA Repository Setup

1. Our next step is to set up remote_child as a child of primary_root. On remote_child, run the following command:

```
# rpkic create_identity remote_child
```

This will produce a file named *remote_child.identity.xml*.

2. We now want to copy this over to primary_root by using rsync.

a. First, copy the file to */usr/share/rpki/publication* on remote_child.

b. Next, on primary_root, run the following command:

```
# rsync rsync://192.168.2.116/rpki/remote_child.identity.xml ./
```

(Replace *192.168.2.116* with remote_child's IP address in the command above.)

This command will copy the child's identity file to the current working directory on primary_root.

c. Now, on primary_root, run the following command:

```
# rpkic configure_child remote_child.identity.xml
```

This will produce a file named *primary_root.remote_child.parent-response.xml*.

3. We will copy this file over to remote_child.

   a. To do this, first (on primary_root) copy the file to /usr/share/rpki/*publication*.

   b. Next, on remote_child, run the following command:

```
# rsync rsync://192.168.2.115/rpki/primary_root.remote_child.parent-
response.xml ./
```

(Replace the IP address with the appropriate one for primary_root in the command above.)

This command will copy the response to the current working directory on remote_child.

   c. With this file, we now run the following command on remote_child:

```
# rpkic configure_parent primary_root.remote_child.parent-response.xml
```

This will produce a file named *remote_child.primary_root.repository-request.xml*.

4. We will copy this file to primary_root with rsync.

   a. To do this, on remote_child, copy the file to */usr/share/rpki/publication*.

   b. Then, on primary_root, run the following command:

```
# rsync rsync://192.168.2.116/rpki/remote_child.primary_root.repository-
request.xml ./
```

(Replace the IP address in the command above with remote_child's IP address).

This will copy the file to the current working directory.

   c. Now, on primary_root, we run the following command:

```
# rpkic configure_publication_client
remote_child.primary_root.repository-request.xml
```

This will produce a file named *remote_child.repository-response.xml*.

5. We will copy this file to the remote_child by using rsync.

   a. On primary_root, copy the file to */usr/share/rpki/publication*.

b. Then, on remote_child, run the following command:

```
# rsync rsync://192.168.2.115/rpki/remote_child.repository-response.xml
./
```

(Replace the IP address as necessary in the command above.)

This will copy the file to the current working directory.

c. Now, on remote_child, we run the following command:

```
# rpkic configure_repository remote_child.repository-response.xml
```

2.2.1.7 Run rcynic to Update Root and Child CA Repositories

This will complete the parent-child setup between primary_root and remote_child. Before verifying, we run the following commands on both VMs:

```
# rpkic force_publication
# rpkic force_run_now
# rpkic synchronize
# sudo -u rpki python /usr/bin/rcynic-cron
```

This should force both VMs to fully update everything, including running rcynic. At this point, you should verify that primary_root shows up as a parent on remote_child's GUI, and that remote_child shows up as a child on primary_root's GUI. Now, we can assign resources. On primary_root's GUI, assign some resources to remote_child. Given enough time, remote_child should update its GUI to reflect that it has been assigned resources under the resources header on the GUI.

2.2.1.8 Adding Resources

When adding resources using the GUI, run the following commands to ensure that rcynic runs to update the repository:

```
# rpkic force_run_now
# rpkic synchronize
# sudo -u rpki python /usr/bin/rcynic-cron
```

2.3 BGP-SRx Software Suite

BGP Secure Routing Extension (BGP-SRx) is an open-source reference implementation and research platform for investigating emerging BGP security extensions and supporting protocols, such as RPKI Origin Validation and Border Gateway Protocol Security (BGPsec) Path Validation [NIST BGP-SRx].

For the latest installation information, please use the Quick Install Guide: https://bgpsrx.antd.nist.gov/bgpsrx/documents/SRxSoftwareSuite-5.0-QuickInstallGuide.pdf.

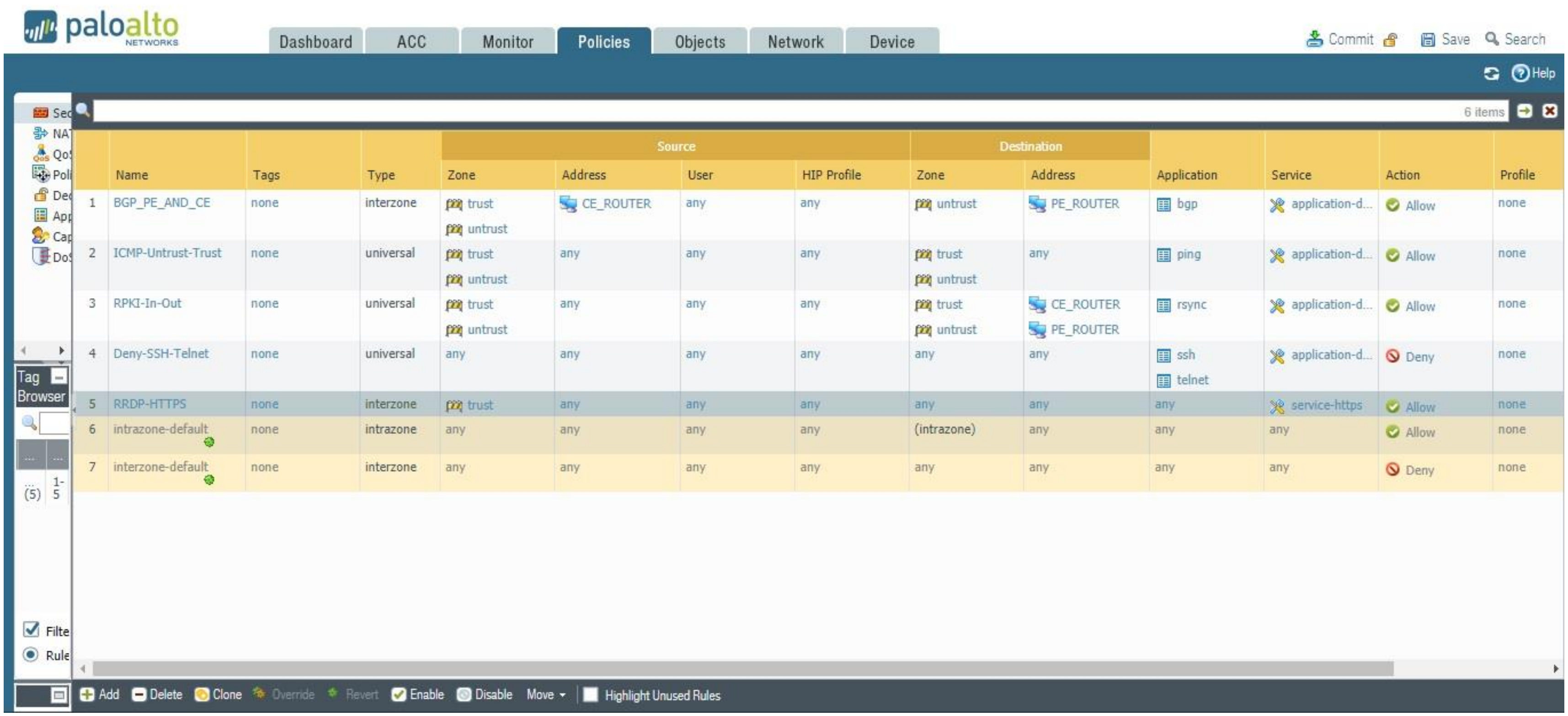
2.4 Firewalls

The firewall used for the lab build is the Palo Alto Next Generation Firewall. The firewall provides protection against known and unknown threats. In this deployment, only ports and connections necessary for the build are configured. All other ports and connections are denied.

System requirements: Palo Alto PA-5060 Next Generation Firewall running Version 7.1.10 software.

The configuration shown in Figure 2-1 addressed all ports that are allowed by the firewall. Ports that are allowed by the firewall are BGP, rsync, and RPKI Repository Delta Protocol (RRDP). All other ports are denied by the firewall. Figure 2-1 depicts the firewall rules.

Figure 2-1 Palo Alto Firewall Configuration



2.5 Test Harness Topology Configuration

The configurations provided in this section are the configurations that are used on each of the routers when operating in the test harness environment architecture provided in Figure 1-1 in Section 1.2. Initially, Cisco routers were used as routers RTR 1-1, RTR 2-1, and RTR 2-2 in that architecture to perform the functional tests. The same tests were then repeated, replacing the Cisco routers with Juniper routers as RTR 1-1, RTR 2-1, and RTR 2-2.

The systems and operating software used for the Cisco routers are as follows:

- Cisco 7206 running *c7200p-adventerprisrk9-mz.152-4.s7.bin*, with a minimum of 4-gigabit Ethernet (GbE) ports. Routers AS 65500 (RTR 2-1) and AS 65501 (RTR 1-1) use this system and OS.
- Cisco 4331 running *ISR4300-universalk9.16.03.04.SPA.bin*, with a minimum of 4 GbE ports. Router AS 65504A (RTR 2-2) uses this system and OS.

All Juniper routers have the following requirements: Juniper MX80 running on Juniper Operating System (JUNOS) 15.1R6.7, with a minimum of 4 GbE ports. Routers AS 65500 (RTR 2-2), AS 65503-J (RTR 2-1), and AS 65505 (RTR 1-1) use this system and OS.

The BGP-SRx Software Suite traffic generators can run on a CentOS Linux system with minimum requirements.

2.5.1 RTR 1-1 Configuration – Cisco

RTR 1-1 acts as an exterior border gateway protocol (eBGP) router receiving eBGP routes from BIO-1, as depicted in Figure 1-1. It updates its interior border gateway protocol (iBGP) peer, BIO-2, with iBGP updates. VRP data is provided to RTR 1-1 by the RPKI validator.

```
hostname AS65501
!
interface GigabitEthernet0/1
 ip address 10.90.90.1 255.255.255.0
 ipv6 address FD00:F:F:1::1/64
!
interface FastEthernet0/2
 description VLAN1
 ip address 192.168.1.2 255.255.255.0
```

```
!
interface GigabitEthernet0/2
 ip address x.x.x.x 255.255.255.252  #Actual IP address to CenturyLink removed.
!
interface GigabitEthernet0/3
 ip address y.y.y.y 255.255.255.248  #Actual IP address to CenturyLink removed.
ipv6 address FD15:F:F:1::1/64

!
router bgp 65501
 bgp log-neighbor-changes
 bgp rpki server tcp 192.168.1.52 port 8282 refresh 5
 neighbor 10.90.90.4 remote-as 65501
 neighbor 192.168.1.50 remote-as 65510
 neighbor 192.168.1.51 remote-as 65511
 neighbor 192.168.1.52 remote-as 65501
 neighbor 192.168.1.53 remote-as 65512
 neighbor FD00:F:F:1::3 remote-as 65503
 !
 address-family ipv4
  bgp bestpath prefix-validate allow-invalid
  no neighbor 10.90.90.4 activate
  neighbor 192.168.1.50 activate
  neighbor 192.168.1.51 activate
  neighbor 192.168.1.52 activate
  neighbor 192.168.1.52 send-community both
```

```
  neighbor 192.168.1.52 announce rpki state
  neighbor 192.168.1.53 activate
  no neighbor FD00:F:F:1::3 activate
 exit-address-family
 !
 address-family ipv6
  redistribute connected
  neighbor FD00:F:F:1::3 activate
 exit-address-family
!
ip prefix-list WAN-OUT seq 10 permit 65.118.221.8/29
!
route-map rpki permit 10
 match rpki invalid
 set local-preference 100
!
route-map RPKI-TEST permit 10
 match ip address prefix-list WAN-OUT
 set community 13698023
!
end
```

2.5.2 RTR 2-1 Configuration – Cisco

RTR 2-1 acts as an eBGP router receiving eBGP routes from BIO-0, and as an iBGP peer providing updates to RTR 2-2, as depicted in Figure 1-1. RTR 2-1 updates another iBGP peer, BIO-2, with iBGP updates. VRP data is provided to RTR 1-1 by the RPKI validator.

```
hostname AS65500
!
interface Loopback1
 ip address 10.100.0.1 255.255.0.0
 ipv6 address 2010:10:10:10::1/64
!
interface GigabitEthernet0/1
 ip address 10.90.90.10 255.255.255.0
  ipv6 address FD00:F:F:1::10/64
!
interface FastEthernet0/2
 ip address 192.168.1.4 255.255.255.0
!
interface GigabitEthernet0/2
 ip address 10.99.99.21 255.255.255.252
!
interface GigabitEthernet0/3
 description VLAN8
!
router bgp 65500
 bgp log-neighbor-changes
 bgp rpki server tcp 192.168.1.52 port 8282 refresh 5
```

```
 bgp rpki server tcp 192.168.1.53 port 8282 refresh 5
 neighbor 192.168.1.5 remote-as 65500
 neighbor 192.168.1.50 remote-as 65510
 neighbor 192.168.1.51 remote-as 65511
 neighbor 192.168.1.52 remote-as 65500
 neighbor 192.168.1.53 remote-as 65513
 !
 address-family ipv4
  bgp bestpath prefix-validate allow-invalid
  redistribute connected
  neighbor 192.168.1.5 activate
  neighbor 192.168.1.5 send-community both
  neighbor 192.168.1.5 announce rpki state
  neighbor 192.168.1.50 activate
  neighbor 192.168.1.51 activate
  neighbor 192.168.1.52 activate
  neighbor 192.168.1.52 send-community both
  neighbor 192.168.1.52 announce rpki state
  neighbor 192.168.1.53 activate
 exit-address-family
!
route-map 10 permit 10
!
end
```

2.5.3 RTR 2-2 Configuration – Cisco

RTR 2-2 acts as an iBGP router receiving iBGP routes from RTR 2-1, and as an eBGP peer providing updates to BIO-6, as depicted in Figure 1-1.

```
version 16.3
!
hostname AS65504A
!
interface GigabitEthernet0/0/0
 description VLNA5
 ip address 10.40.0.1 255.255.255.0
  ipv6 address FD34:F:F:1::4/64
!
interface GigabitEthernet0/0/1
 description VLN6
 ip address 10.99.99.18 255.255.255.252
ipv6 address FD24:F:F:1::4/64
!
interface GigabitEthernet0/0/2
 ip address 192.168.1.5 255.255.255.0
  ipv6 address 2004:4444:4444:4444::4/64
!
router bgp 65500
 bgp log-neighbor-changes
 bgp rpki server tcp 192.168.1.53 port 8282 refresh 5
 bgp rpki server tcp 192.168.1.52 port 8282 refresh 5
 neighbor 192.168.1.4 remote-as 65500
```

```
 neighbor 192.168.1.53 remote-as 65513
 !
 address-family ipv4
  neighbor 192.168.1.4 activate
  neighbor 192.168.1.4 send-community both
  neighbor 192.168.1.4 announce rpki state
  neighbor 192.168.1.53 activate
 exit-address-family
!
route-map NO-EXPORT permit 10
 set community no-export
!
end
```

2.5.4 RTR 1-1 Configuration – Juniper

RTR 1-1 acts as an eBGP router receiving eBGP routes from BIO-1, as depicted in Figure 1-1. RTR 1-1 updates its iBGP peer, BIO-2, with iBGP updates. VRP data is provided to it by the RPKI validator.

```
set system host-name AS65501
set system login user nccoe uid 2000
set system login user nccoe class read-only
set system login user nccoe authentication encrypted-password
"$5$8.Yu28ng$LbcoMQ9uqDO3.U4VaiG4bg5fWMeaMYAJjr09Aniu8c7"
set interfaces ge-1/3/0 unit 0 family inet address 192.168.1.12/24
set interfaces ge-1/3/1 unit 0 family inet
set interfaces ge-1/3/2 unit 0 family inet
set interfaces ge-1/3/3 unit 0 family inet
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
set routing-options autonomous-system 65501
```

```
set routing-options validation group cache session 192.168.1.52 refresh-time 5
set routing-options validation group cache session 192.168.1.52 port 8282
set protocols bgp group external-as65511 type external
set protocols bgp group external-as65511 import validation
set protocols bgp group external-as65511 export allow-direct
set protocols bgp group external-as65511 peer-as 65511
set protocols bgp group external-as65511 neighbor 192.168.1.51
set protocols bgp group external-as65510 type external
set protocols bgp group external-as65510 import validation
set protocols bgp group external-as65510 export allow-direct
set protocols bgp group external-as65510 peer-as 65510
set protocols bgp group external-as65510 neighbor 192.168.1.50
set protocols bgp group internal-as65501 type internal
set protocols bgp group internal-as65501 neighbor 192.168.1.52
set protocols bgp group external-as65512 type external
set protocols bgp group external-as65512 import validation
set protocols bgp group external-as65512 export allow-direct
set protocols bgp group external-as65512 peer-as 65512
set protocols bgp group external-as65512 neighbor 192.168.1.53
set policy-options policy-statement allow-all from route-filter 0.0.0.0/0
orlonger
set policy-options policy-statement allow-all then accept
set policy-options policy-statement allow-direct term default from protocol
direct
set policy-options policy-statement allow-direct term default then accept
set policy-options policy-statement validation term valid from protocol bgp
set policy-options policy-statement validation term valid from validation-
database valid
```

```
set policy-options policy-statement validation term valid then local-preference 110

set policy-options policy-statement validation term valid then validation-state valid

set policy-options policy-statement validation term valid then community add origin-validation-state-valid

set policy-options policy-statement validation term valid then accept

set policy-options policy-statement validation term invalid from protocol bgp

set policy-options policy-statement validation term invalid from validation-database invalid

set policy-options policy-statement validation term invalid then local-preference 90

set policy-options policy-statement validation term invalid then validation-state invalid

set policy-options policy-statement validation term invalid then community add origin-validation-state-invalid

set policy-options policy-statement validation term invalid then accept

set policy-options policy-statement validation term unknown from protocol bgp

set policy-options policy-statement validation term unknown then validation-state unknown

set policy-options policy-statement validation term unknown then community add origin-validation-state-unknown

set policy-options policy-statement validation term unknown then accept

set policy-options community origin-validation-state-invalid members 0x4300:2

set policy-options community origin-validation-state-unknown members 0x4300:1

set policy-options community origin-validation-state-valid members 0x4300:0
```

2.5.5 RTR 2-1 Configuration – Juniper

RTR 2-1 acts as an eBGP router receiving eBGP routes from BIO-0, and as an iBGP peer providing updates to RTR 2-2, as depicted in Figure 1-1. It updates another iBGP peer, BIO-2, with iBGP updates. VRP data is provided to RTR 2-1 by the RPKI validator.

```
set system host-name AS65500-J
set interfaces ge-1/3/0 unit 0 family inet
set interfaces ge-1/3/1 unit 0 family inet address 192.168.1.14/24
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
set routing-options autonomous-system 65500
set routing-options validation traceoptions file rpki-trace
set routing-options validation traceoptions flag all
deactivate routing-options validation traceoptions
set routing-options validation group cache session 192.168.1.52 refresh-time 5
set routing-options validation group cache session 192.168.1.52 port 8282
set protocols bgp group external-as65511 type external
set protocols bgp group external-as65511 import validation
set protocols bgp group external-as65511 export allow-direct
set protocols bgp group external-as65511 peer-as 65511
set protocols bgp group external-as65511 neighbor 192.168.1.51
set protocols bgp group external-as65510 type external
set protocols bgp group external-as65510 import validation
set protocols bgp group external-as65510 export allow-direct
set protocols bgp group external-as65510 peer-as 65510
set protocols bgp group external-as65510 neighbor 192.168.1.50
set protocols bgp group internal-as65500 type internal
set protocols bgp group internal-as65500 neighbor 192.168.1.52
```

```
set policy-options policy-statement allow-all from route-filter 0.0.0.0/0
orlonger

set policy-options policy-statement allow-all then accept

set policy-options policy-statement allow-direct term default from protocol
direct

set policy-options policy-statement allow-direct term default then accept

set policy-options policy-statement validation term valid from protocol bgp

set policy-options policy-statement validation term valid from validation-
database valid

set policy-options policy-statement validation term valid then local-preference
110

set policy-options policy-statement validation term valid then validation-state
valid

set policy-options policy-statement validation term valid then community add
origin-validation-state-valid

set policy-options policy-statement validation term valid then accept

set policy-options policy-statement validation term invalid from protocol bgp

set policy-options policy-statement validation term invalid from validation-
database invalid

set policy-options policy-statement validation term invalid then local-
preference 90

set policy-options policy-statement validation term invalid then validation-
state invalid

set policy-options policy-statement validation term invalid then community add
origin-validation-state-invalid

set policy-options policy-statement validation term invalid then accept

set policy-options policy-statement validation term unknown from protocol bgp

set policy-options policy-statement validation term unknown then validation-
state unknown

set policy-options policy-statement validation term unknown then community add
origin-validation-state-unknown

set policy-options policy-statement validation term unknown then accept
```

```
set policy-options community origin-validation-state-invalid members 0x4300:0:2
set policy-options community origin-validation-state-unknown members 0x4300:0:1
set policy-options community origin-validation-state-valid members 0x4300:0:0
```

2.5.6 RTR 2-2 Configuration – Juniper

RTR 2-2 acts as an iBGP router receiving iBGP routes from RTR 2-1, and as an eBGP peer providing updates to BIO-6, as depicted in Figure 1-1.

```
set system host-name AS65500
set interfaces ge-1/3/0 unit 0 family inet address 192.168.1.15/24
set interfaces ge-1/3/1 unit 0
set interfaces ge-1/3/2 unit 0
set interfaces ge-1/3/3 unit 0
set interfaces lo0 unit 0 family inet
set routing-options autonomous-system 65500
set routing-options validation group cache session 192.168.1.52 refresh-time 5
set routing-options validation group cache session 192.168.1.52 port 8282
set routing-options validation group cache session 192.168.1.53 refresh-time 5
set routing-options validation group cache session 192.168.1.53 port 8282
set protocols bgp group internal-as65500 type internal
set protocols bgp group internal-as65500 neighbor 192.168.1.14
set protocols bgp group external-as65513 type external
set protocols bgp group external-as65513 import validation
set protocols bgp group external-as65513 export allow-direct
set protocols bgp group external-as65513 peer-as 65513
set protocols bgp group external-as65513 neighbor 192.168.1.53
set policy-options policy-statement allow-all from route-filter 0.0.0.0/0
orlonger
set policy-options policy-statement allow-all then accept
```

```
set policy-options policy-statement allow-direct term default from protocol direct
set policy-options policy-statement allow-direct term default then accept
set policy-options policy-statement validation term valid from protocol bgp
set policy-options policy-statement validation term valid from validation-database valid
set policy-options policy-statement validation term valid then local-preference 110
set policy-options policy-statement validation term valid then validation-state valid
set policy-options policy-statement validation term valid then community add origin-validation-state-valid
set policy-options policy-statement validation term valid then accept
set policy-options policy-statement validation term invalid from protocol bgp
set policy-options policy-statement validation term invalid from validation-database invalid
set policy-options policy-statement validation term invalid then local-preference 90
set policy-options policy-statement validation term invalid then validation-state invalid
set policy-options policy-statement validation term invalid then community add origin-validation-state-invalid
set policy-options policy-statement validation term invalid then accept
set policy-options policy-statement validation term unknown from protocol bgp
set policy-options policy-statement validation term unknown then validation-state unknown
set policy-options policy-statement validation term unknown then community add origin-validation-state-unknown
set policy-options policy-statement validation term unknown then accept
set policy-options community origin-validation-state-invalid members 0x4300:2
set policy-options community origin-validation-state-invalid members 0x43:100:2
set policy-options community origin-validation-state-unknown members 0x4300:1
```

```
set policy-options community origin-validation-state-valid members 0x4300:0
```

2.5.7 Traffic Generator BIO Configuration

```
ski_file    = "/var/lib/key-volt/ski-list.txt";
ski_key_loc = "/var/lib/key-volt/";
preload_eckey = false;
mode = "BGP";
max = 0;
only_extended_length = true;
session = (
{
    disconnect = 0;
    ext_msg_cap     = true;
    ext_msg_liberal = true;
    bgpsec_v4_snd = false;
    bgpsec_v4_rcv  = false;
    bgpsec_v6_snd = false;
  bgpsec_v6_rcv  = false;    update = (
                );
    incl_global_updates = true;
    algo_id = 1;
    signature_generation = "BIO";
    null_signature_mode = "FAKE";
    fake_signature           = "1BADBEEFDEADFEED" "2BADBEEFDEADFEED"
                                           "3BADBEEFDEADFEED" "4BADBEEFDEADFEED"
                                           "5BADBEEFDEADFEED" "6BADBEEFDEADFEED"
                                           "7BADBEEFDEADFEED" "8BADBEEFDEADFEED"
                                           "ABADBEEFFACE";
    fake_ski                     = "0102030405060708" "090A0B0C0D0E0F10"
                                          "11121314";
    printOnSend = {
```

```
          update       = true;
        };

        printOnReceive = {
          update.      = true;
          notification = true;
          unknown    = true;
        };
        printSimple     = true;
        printPollLoop  = false;
        printOnInvalid = false;
      }
    );
    update = (
             );
```

2.5.7.1 AS – Peer Configuration: BIO-0 (AS 65510) – RTR-1-1 (AS 65501)

```
        asn            = 65510;
        bgp_ident  = "192.168.1.50";
        hold_timer = 180;

        peer_asn   = 65501;
        # For CISCO replace x with 2, For JUNIPER replace x with 12
        peer_ip     = "192.168.1.x";
        peer_port  = 179;
```

2.5.7.2 AS – Peer Configuration: BIO-0 (AS 65510) – RTR-2-1 (AS 65500)

```
    asn            = 65510;
        bgp_ident  = "192.168.1.50";
        hold_timer = 180;

        peer_asn   = 65500;
```

```
        # For CISCO replace x with 4, For JUNIPER replace x with 14
        peer_ip     = "192.168.1.x";
        peer_port  = 179;
```

2.5.7.3 AS – Peer Configuration: BIO-1 (AS 65511) – RTR-1-1 (AS 65501)

```
    asn             = 65511;
        bgp_ident  = "192.168.1.51";
        hold_timer = 180;

        peer_asn   = 65500;
        # For CISCO replace x with 2, For JUNIPER replace x with 12
        peer_ip     = "192.168.1.x";
        peer_port  = 179;
```

2.5.7.4 AS – Peer Configuration: BIO-1 (AS 65511) – RTR-2-1 (AS 65500)

```
        asn             = 65511;
        bgp_ident  = "192.168.1.51";
        hold_timer = 180;

        peer_asn   = 65500;
        # For CISCO replace x with 4, For JUNIPER replace x with 14
        peer_ip     = "192.168.1.x";
        peer_port  = 179;
```

2.5.7.5 AS – Peer Configuration: BIO-2 (AS 65501) – RTR-1-1 (AS 65501)

```
        asn             = 65501;
        bgp_ident  = "192.168.1.52";
        hold_timer = 180;

        peer_asn   = 65501;
        # For CISCO replace x with 2, For JUNIPER replace x with 12
        peer_ip     = "192.168.1.x";
        peer_port  = 179;
```

2.5.7.6 AS – Peer Configuration: BIO-3 (AS 65500) – RTR-2-1 (AS 65500)

```
asn           = 65500;
bgp_ident  = "192.168.1.52";
hold_timer = 180;

peer_asn   = 65500;
# For CISCO replace x with 4, For JUNIPER replace x with 14
peer_ip     = "192.168.1.x";
peer_port  = 179;
```

2.5.7.7 AS – Peer Configuration: BIO-5 (AS 65512) – RTR-1-1 (AS 65500)

```
asn           = 65512;
bgp_ident  = "192.168.1.53";
hold_timer = 180;

peer_asn   = 65501;
# For CISCO replace x with 2, For JUNIPER replace x with 12
peer_ip     = "192.168.1.x";
peer_port  = 179;
```

2.5.7.8 AS – Peer Configuration: BIO-6 (AS 65513) – RTR-1-1 (AS 65513)

```
asn           = 65513;
bgp_ident  = "192.168.1.53";
hold_timer = 180;

peer_asn   = 65500;
# For CISCO replace x with 4, For JUNIPER replace x with 14
peer_ip     = "192.168.1.x";
peer_port  = 179;
```

2.6 Live Data Configuration

The configurations provided in this section are the configurations that are used on each of the routers when operating in the live data environment architecture shown in Figure 1-2. Live BGP data and RPKI data can be retrieved in this environment. The architecture is organized into eight separate networks, each of which is designed to represent a different AS.

The systems and operating software used for the Cisco routers are as follows:

- Cisco 7206 running *c7200p-adventerprisrk9-mz.152-4.s7.bin*, with a minimum of 4 GbE ports. Routers AS 65500, AS 65501, and AS 65503 use this system and OS.
- Cisco 4331 running *ISR4300-universalk9.16.03.04.SPA.bin*, with a minimum of 4 GbE ports. Routers AS 65504A and AS 65504B use this system and OS.
- Cisco 2921 running *c2900-universalk9-mz-SPA.152-4.M6.bin*, with a minimum of 4 GbE ports. Routers AS 65507 and AS 65508 use this system and OS.
- Cisco Internetwork Operating System (IOS) XRv 9000 router Version 6.4.1 running on VMware ESXi using the *xrv9k-fullk9-x.vrr-6.4.1.ova* file.

All Juniper routers have the following requirements: Juniper MX80 running on JUNOS 15.1R6.7, with a minimum of 4 GbE ports. Routers AS 65502 and AS 65505 use this system and OS.

RPKI validators and repositories are configured based on Section 2.1 and Section 2.2. Live ROV data is retrieved from the five trust anchors, and lab ROA data is retrieved from the lab delegated model of the local RPKI repository.

Note: Real IP addresses and AS numbers were removed from the configuration.

2.6.1 CenturyLink Configuration Router AS 65501 – Cisco

To receive a full BGP route table, CenturyLink provided a physical link connecting the NCCoE lab with an eBGP peering. The configuration below illustrates the eBGP peering. An additional configuration for this router, related to the lab build, is provided in Section 2.5.3.

```
version 15.2
!
hostname AS65501
!
ipv6 unicast-routing
ipv6 cef
```

```
!
interface GigabitEthernet0/1
 ip address 10.90.90.1 255.255.255.0
ipv6 address FD00:F:F:1::1/64
!
interface FastEthernet0/2
 description VLAN1
 ip address 192.168.1.2 255.255.255.0
!
interface GigabitEthernet0/2
 ip address a.a.a.a 255.255.255.252
!
interface GigabitEthernet0/3
 ip address c.c.c.c 255.255.255.248

ipv6 address FD15:F:F:1::1/64
!
router bgp aaa
 bgp log-neighbor-changes
 neighbor a.a.a.b remote-as bbb
!
 address-family ipv4
  network c.c.c.d mask 255.255.255.248
  neighbor a.a.a.b activate
  neighbor a.a.a.b send-community
  neighbor a.a.a.b soft-reconfiguration inbound
```

```
  neighbor a.a.a.b route-map RPKI-TEST out
 exit-address-family
!
ip prefix-list WAN-OUT seq 10 permit c.c.c.d/29
ipv6 router rip proc1
!
route-map rpki permit 10
 match rpki invalid
 set local-preference 100
!
route-map RPKI-TEST permit 10
 match ip address prefix-list WAN-OUT
 set community 13698023
!
end
```

2.6.2 Router AS 65500 Configuration – Cisco

Router AS 65500 represents an ISP. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peers.

```
hostname AS65500
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface Loopback1
 ip address 10.10.0.1 255.255.0.0
```

```
 ipv6 address FD10:10:10:10::1/64
 ipv6 rip proc1 enable
!
interface GigabitEthernet0/1
 ipv6 address FD00:F:F:1::1/64
 ipv6 rip proc1 enable
!
interface FastEthernet0/2
 description VLAN1
 ip address 192.168.1.2 255.255.255.0
 ipv6 address FD01:F:F:1::2/64
 ipv6 rip proc1 enable
!
interface GigabitEthernet0/2
 ip address a.a.a.a 255.255.255.252
!
interface GigabitEthernet0/3
 ip address c.c.c.c 255.255.255.248
 ipv6 address FD15:F:F:1::1/64
!
router rip
 version 2
 network 10.0.0.0
 network 192.168.1.0
 no auto-summary
!
```

```
router bgp aaa
 bgp log-neighbor-changes
 neighbor a.a.a.b remote-as bbb
 !
 address-family ipv4
  network c.c.c.d mask 255.255.255.248
  neighbor a.a.a.b activate
  neighbor a.a.a.b send-community
  neighbor a.a.a.b soft-reconfiguration inbound
  neighbor a.a.a.b route-map RPKI-TEST out
 exit-address-family
!
ip route 10.20.0.0 255.255.0.0 192.168.1.3
ip route 10.30.0.0 255.255.0.0 192.168.1.3
ip route 10.40.0.0 255.255.0.0 192.168.1.3
ip route 10.50.0.0 255.255.0.0 192.168.1.3
ip route 10.70.0.0 255.255.0.0 192.168.1.3
ip route 10.80.0.0 255.255.0.0 192.168.1.3
ip route 10.90.90.0 255.255.255.0 192.168.1.3
ip route 10.97.74.0 255.255.255.0 192.178.1.1
ip route 10.99.99.0 255.255.255.0 192.168.1.3
!
ip prefix-list WAN-OUT seq 10 permit c.c.c.d /29
ipv6 router rip proc1
!
route-map rpki permit 10
```

```
 match rpki invalid
 set local-preference 100
!
route-map RPKI-TEST permit 10
 match ip address prefix-list WAN-OUT
 set community 13698023
!
end
```

2.6.3 Router 65501 Configuration – Cisco

Router AS 65501 represents an ISP. As indicated in Section 2.5.1, this router peers with the CenturyLink router to receive a full BGP routing table. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peers. It is the gateway for all devices in the lab, allowing ROAs from RIRs to be retrieved by RPKI validators. It also peers with stub AS A65505.

```
hostname AS65501
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface Loopback1
 ip address 10.10.0.1 255.255.0.0
 ipv6 address FD10:10:10:10::1/64
 ipv6 rip proc1 enable
!
interface GigabitEthernet0/1
 ipv6 address FD00:F:F:1::1/64
 ipv6 rip proc1 enable
```

```
!
interface FastEthernet0/2
 ip address 192.168.1.2 255.255.255.0
 ipv6 address FD01:F:F:1::2/64
 ipv6 rip proc1 enable
!
interface GigabitEthernet0/2
 ip address a.a.a.a 255.255.255.252
!
interface GigabitEthernet0/3
 ip address c.c.c.c 255.255.255.248
 ipv6 address FD15:F:F:1::1/64
!
router rip
 version 2
 network 10.0.0.0
 network 192.168.1.0
 no auto-summary
!
router bgp aaa
 bgp log-neighbor-changes
 neighbor a.a.a.b remote-as bbb
 !
 address-family ipv4
  network c.c.c.d mask 255.255.255.248
  neighbor a.a.a.b activate
```

```
  neighbor a.a.a.b send-community
  neighbor a.a.a.b soft-reconfiguration inbound
  neighbor a.a.a.b route-map RPKI-TEST out
 exit-address-family
!
ip route 10.20.0.0 255.255.0.0 192.168.1.3
ip route 10.30.0.0 255.255.0.0 192.168.1.3
ip route 10.40.0.0 255.255.0.0 192.168.1.3
ip route 10.50.0.0 255.255.0.0 192.168.1.3
ip route 10.70.0.0 255.255.0.0 192.168.1.3
ip route 10.80.0.0 255.255.0.0 192.168.1.3
ip route 10.90.90.0 255.255.255.0 192.168.1.3
ip route 10.97.74.0 255.255.255.0 192.178.1.1
ip route 10.99.99.0 255.255.255.0 192.168.1.3
!
ip prefix-list WAN-OUT seq 10 permit c.c.c.d /29
ipv6 router rip proc1
!
route-map rpki permit 10
 match rpki invalid
 set local-preference 100
!
route-map RPKI-TEST permit 10
 match ip address prefix-list WAN-OUT
 set community 13698023
!
```

```
end
```

2.6.4 Router AS 65502 Configuration – Juniper

Router AS 65502 represents an ISP using a Juniper router. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peers. It also provides eBGP routes to stub AS 65504.

```
set system host-name AS65502
set interfaces ge-1/3/0 unit 0 family inet address 10.90.90.2/24
set interfaces ge-1/3/0 unit 0 family inet6 address fd00:f:f:1::2/64
set interfaces ge-1/3/1 unit 0 family inet address 10.99.99.17/30
set interfaces ge-1/3/1 unit 0 family inet6 address fd24:f:f:1::2/64
set interfaces ge-1/3/2 unit 0 family inet address 10.99.99.25/30
set interfaces ge-1/3/2 unit 0 family inet6 address fd25:f:f:1::2/64
set interfaces ge-1/3/3 unit 0 family inet address 10.20.0.1/16
set interfaces ge-1/3/3 unit 0 family inet6 address 2020:2020:2020:1::2/64
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
set routing-options validation group cache session 192.168.1.146 port 8282
set policy-options policy-statement allow-all from route-filter 0.0.0.0/0
orlonger
set policy-options policy-statement allow-all then accept
set routing-instances rpki instance-type virtual-router
set routing-instances rpki interface ge-1/3/0.0
set routing-instances rpki interface ge-1/3/1.0
set routing-instances rpki interface ge-1/3/2.0
set routing-instances rpki interface ge-1/3/3.0
set routing-instances rpki interface lo0.1
set routing-instances rpki routing-options router-id 2.2.2.2
set routing-instances rpki routing-options autonomous-system 65502
```

```
set routing-instances rpki protocols bgp group external-as65500 type external

set routing-instances rpki protocols bgp group external-as65500 import allow-all

set routing-instances rpki protocols bgp group external-as65500 export allow-all

set routing-instances rpki protocols bgp group external-as65500 peer-as 65500

set routing-instances rpki protocols bgp group external-as65500 neighbor 10.90.90.10

set routing-instances rpki protocols bgp group external-as65500 neighbor fd00:f:f:1::10

set routing-instances rpki protocols bgp group external-as65501 type external

set routing-instances rpki protocols bgp group external-as65501 import allow-all

set routing-instances rpki protocols bgp group external-as65501 export allow-all

set routing-instances rpki protocols bgp group external-as65501 peer-as 65501

set routing-instances rpki protocols bgp group external-as65501 neighbor 10.90.90.1

set routing-instances rpki protocols bgp group external-as65501 neighbor fd00:f:f:1::1

set routing-instances rpki protocols bgp group external-as65503 type external

set routing-instances rpki protocols bgp group external-as65503 import allow-all

set routing-instances rpki protocols bgp group external-as65503 export allow-all

set routing-instances rpki protocols bgp group external-as65503 peer-as 65503

set routing-instances rpki protocols bgp group external-as65503 neighbor 10.90.90.3

set routing-instances rpki protocols bgp group external-as65503 neighbor fd00:f:f:1::3

set routing-instances rpki protocols bgp group external-as65505 type external

set routing-instances rpki protocols bgp group external-as65505 import allow-all
```

```
set routing-instances rpki protocols bgp group external-as65505 export allow-
all

set routing-instances rpki protocols bgp group external-as65505 peer-as 65505

set routing-instances rpki protocols bgp group external-as65505 neighbor
fd25:f:f:1::5

set routing-instances rpki protocols bgp group external-as65505 neighbor
10.99.99.26

set routing-instances rpki protocols bgp group external-as65504 type external

set routing-instances rpki protocols bgp group external-as65504 import allow-
all

set routing-instances rpki protocols bgp group external-as65504 export allow-
all

set routing-instances rpki protocols bgp group external-as65504 peer-as 65504

set routing-instances rpki protocols bgp group external-as65504 neighbor
10.99.99.18

set routing-instances rpki protocols bgp group external-as65504 neighbor
fd24:f:f:1::4
```

2.6.5 Router AS 65503 Configuration – Cisco

Router AS 65503 represents an ISP without ROV capabilities. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peers without performing BGP origin validation. This router peers with two transit routers, AS 65500 and AS 65502, as well as two stub ASes, AS 65504 and AS 65507.

```
hostname AS65503

!

ip cef

ipv6 unicast-routing

ipv6 cef

!

interface Loopback1

 ip address 10.30.0.1 255.255.0.0

 ipv6 address 2003:3333:3333:3333::1/64
```

```
!
interface GigabitEthernet0/1
 ip address 10.90.90.3 255.255.255.0
 ipv6 address FD00:F:F:1::3/64
!
interface FastEthernet0/2
 ip address 192.168.1.251 255.255.255.0
!
interface GigabitEthernet0/2
 ip address 10.99.99.13 255.255.255.252
!
interface GigabitEthernet0/3
 description VLAN7
 ip address 10.99.99.21 255.255.255.252
 ipv6 address FD37:F:F:1::1/64
!
router bgp 65503
 bgp log-neighbor-changes
 bgp rpki server tcp 192.168.1.146 port 8282 refresh 10
 neighbor 10.90.90.1 remote-as 65501
 neighbor 10.90.90.2 remote-as 65502
 neighbor 10.90.90.10 remote-as 65500
 neighbor 10.99.99.14 remote-as 65504
 neighbor 10.99.99.22 remote-as 65507
 neighbor FD00:F:F:1::1 remote-as 65501
 neighbor FD00:F:F:1::2 remote-as 65502
```

```
neighbor FD00:F:F:1::10 remote-as 65500
neighbor FD34:F:F:1::4 remote-as 65504
neighbor FD34:F:F:1::7 remote-as 65507
!
address-family ipv4
 redistribute connected
 redistribute static
 neighbor 10.90.90.1 activate
 neighbor 10.90.90.2 activate
 neighbor 10.90.90.10 activate
 neighbor 10.99.99.14 activate
 neighbor 10.99.99.22 activate
 no neighbor FD00:F:F:1::1 activate
 no neighbor FD00:F:F:1::2 activate
 no neighbor FD00:F:F:1::10 activate
 no neighbor FD34:F:F:1::4 activate
 no neighbor FD34:F:F:1::7 activate
exit-address-family
!
address-family ipv6
 redistribute connected
 neighbor FD00:F:F:1::1 activate
 neighbor FD00:F:F:1::2 activate
 neighbor FD00:F:F:1::10 activate
 neighbor FD34:F:F:1::4 activate
exit-address-family
```

```
!
ipv6 router rip proc1
!
end
```

2.6.6 Router AS 65504A Configuration – Cisco

Router AS 65504A represents an enterprise edge router for AS 65504. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peer, AS 65502. It peers with Router AS 65504B to exchange iBGP routes.

```
hostname AS65504A
!
ipv6 unicast-routing
!
interface Loopback1
 ip address 10.40.1.1 255.255.255.0
!
interface GigabitEthernet0/0/0
 ip address 10.40.0.1 255.255.255.0
 ipv6 address FD00:F:F:1::40/64
 ipv6 address FD34:F:F:1::4/64
!
interface GigabitEthernet0/0/1
 ip address 10.99.99.18 255.255.255.252
 ipv6 address FD24:F:F:1::4/64
!
interface GigabitEthernet0/0/2
 ip address 10.40.4.1 255.255.255.0
```

```
 ipv6 address 2004:4444:4444:4444::4/64
!
router bgp 65504
 bgp log-neighbor-changes
 neighbor 10.40.0.2 remote-as 65504
 neighbor 10.99.99.17 remote-as 65502
 neighbor FD24:F:F:1::2 remote-as 65502
 !
 address-family ipv4
  redistribute connected
  redistribute static
  no neighbor 10.40.0.2 activate
  neighbor 10.99.99.17 activate
  no neighbor FD24:F:F:1::2 activate
 exit-address-family
 !
 address-family ipv6
  redistribute connected
  neighbor FD24:F:F:1::2 activate
 exit-address-family
!
ip route 10.40.2.0 255.255.255.0 10.40.0.2
!
route-map NO-EXPORT permit 10
 set community no-export
!
```

```
end
```

2.6.7 Router AS 65504B Configuration – Cisco

Router AS 65504B represents an enterprise edge router for AS 65504. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peer, AS 65503. It peers with Router AS 65504A to exchange iBGP routes.

```
hostname AS65504B
!
ipv6 unicast-routing
!
interface Loopback1
 ip address 10.40.2.1 255.255.255.0
 ipv6 address 4040:4040:4040:4242::1/64
!
interface GigabitEthernet0/0/0
 ip address 10.99.99.14 255.255.255.252
 ipv6 address FD34:F:F:1::4/64
!
interface GigabitEthernet0/0/1
 ip address 10.40.0.2 255.255.255.0
 ipv6 address FD40:F:F:1::2/64
!
router bgp 65504
 bgp log-neighbor-changes
 neighbor 10.40.0.1 remote-as 65504
 neighbor 10.99.99.13 remote-as 65503
 neighbor FD34:F:F:1::2 remote-as 65503
```

```
  neighbor FD40:F:F:1::1 remote-as 65504
  !
  address-family ipv4
   redistribute connected
   no neighbor 10.40.0.1 activate
   neighbor 10.99.99.13 activate
   no neighbor FD34:F:F:1::2 activate
   no neighbor FD40:F:F:1::1 activate
  exit-address-family
  !
  address-family ipv6
   redistribute connected
   neighbor FD34:F:F:1::2 activate
   neighbor FD40:F:F:1::1 activate
  exit-address-family
 !
 route-map NO-EXPORT permit 10
  set community no-export
 !
 end
```

2.6.8 Router AS 65505 Configuration – Juniper

Router AS 65505 represents an enterprise edge router. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peers, AS 65501 and AS 65502.

```
set system host-name AS65505
set interfaces ge-1/3/0 unit 0 family inet
```

```
set interfaces ge-1/3/0 unit 0 family inet6
set interfaces ge-1/3/1 unit 0 family inet address 10.99.99.2/30
set interfaces ge-1/3/1 unit 0 family inet6 address fd15:f:f:1::5/64
set interfaces ge-1/3/2 unit 0 family inet address 10.99.99.26/30
set interfaces ge-1/3/2 unit 0 family inet6 address fd25:f:f:1::5/64
set interfaces ge-1/3/3 unit 0 family inet address 10.50.0.1/16
set interfaces ge-1/3/3 unit 0 family inet6 address 5050:5050:5050:1::5/64
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
set routing-options autonomous-system 65505
set routing-options validation group cache session 192.168.1.146 port 8282
set protocols bgp group external-as65501 type external
set protocols bgp group external-as65501 import validation
set protocols bgp group external-as65501 export allow-direct
set protocols bgp group external-as65501 peer-as 65501
set protocols bgp group external-as65501 neighbor 10.99.99.1
set protocols bgp group external-as65501 neighbor fd15:f:f:1::1
set protocols bgp group external-as65502 type external
set protocols bgp group external-as65502 import validation
set protocols bgp group external-as65502 export allow-direct
set protocols bgp group external-as65502 peer-as 65502
set protocols bgp group external-as65502 neighbor 10.99.99.25
set protocols bgp group external-as65502 neighbor fd25:f:f:1::2
set policy-options policy-statement allow-all from route-filter 0.0.0.0/0 
orlonger
set policy-options policy-statement allow-all then accept
set policy-options policy-statement allow-direct term default from protocol 
direct
```

```
set policy-options policy-statement allow-direct term default then accept
set policy-options policy-statement validation term valid from protocol bgp
set policy-options policy-statement validation term valid from validation-database valid
set policy-options policy-statement validation term valid then local-preference 110
set policy-options policy-statement validation term valid then validation-state valid
set policy-options policy-statement validation term valid then accept
set policy-options policy-statement validation term invalid from protocol bgp
set policy-options policy-statement validation term invalid from validation-database invalid
set policy-options policy-statement validation term invalid then local-preference 90
set policy-options policy-statement validation term invalid then validation-state invalid
set policy-options policy-statement validation term invalid then reject
set policy-options policy-statement validation term unknown from protocol bgp
set policy-options policy-statement validation term unknown then validation-state unknown
set policy-options policy-statement validation term unknown then accept
```

2.6.9 Router AS 65507 Configuration – Cisco

Router AS 65507 represents an enterprise edge router for AS 65507. For the lab build, this router originates BGP updates from its own AS and receives and sends routes to and from its eBGP peer, AS 65503.

```
hostname AS65507
!
interface Loopback1
 ip address 10.70.0.1 255.255.0.0
 ipv6 address 7070:7070:7070:7070::1/64
```

```
!
interface GigabitEthernet0/0
 ip address 10.99.99.22 255.255.255.252
 ipv6 address FD37:F:F:1::7/64
!
interface GigabitEthernet0/1
 ip address 172.16.0.1 255.255.0.0
!
router bgp 65507
 bgp log-neighbor-changes
 neighbor 10.99.99.21 remote-as 65503
 neighbor FD37:F:F:1::3 remote-as 65503
 !
 address-family ipv4
  redistribute connected
  neighbor 10.99.99.21 activate
  no neighbor FD37:F:F:1::3 activate
 exit-address-family
 !
 address-family ipv6
  redistribute connected
  neighbor FD37:F:F:1::3 activate
 exit-address-family
!
access-list 23 permit 10.10.10.0 0.0.0.7
ipv6 router rip proc1
```

```
!
end
```

2.6.10 Router AS 65508 Configuration – Cisco

Router AS 65508 represents a hijacker masquerading as an enterprise edge router. For the lab build, this router originates BGP updates for routes that are held by other ASes (i.e., for routes for which it is not authorized to originate updates), in order to demonstrate route hijacks.

```
hostname AS65508
!
ipv6 unicast-routing
ipv6 cef
!
interface Loopback1
 ip address 10.80.0.1 255.255.0.0
 ipv6 address 8080:8080:8080:8080::1/64
!
interface GigabitEthernet0/0
 ip address 10.99.99.30 255.255.255.252
 ipv6 address FD00:F:F:1::61/64
 ipv6 address FD08:F:F:1::8/64
!
interface GigabitEthernet0/1
 ip address 172.16.8.1 255.255.255.0
!
router bgp 65508
 bgp log-neighbor-changes
 neighbor 10.99.99.29 remote-as 65500
```

```
 neighbor FD08:F:F:1::10 remote-as 65500
 !
 address-family ipv4
  redistribute connected
  neighbor 10.99.99.29 activate
  no neighbor FD08:F:F:1::10 activate
 exit-address-family
 !
 address-family ipv6
  redistribute connected
  neighbor FD08:F:F:1::10 activate
 exit-address-family
!
ipv6 router rip proc1
!
end
```

2.6.11 Cisco IOS XRv Router Configuration

The Cisco IOS XRv software was also used to perform many of the functional tests, as many ISPs currently use it in their network environment. The baseline configuration is provided below. Depending on the test case, this router can replace any other router shown in Figure 1-2, in order to properly perform the test.

```
RP/0/RP0/CPU0:ios#sho run
!! IOS XR Configuration version = 6.4.1
!
interface MgmtEth0/RP0/CPU0/0
 ipv4 address 192.168.1.201 255.255.255.0
 ipv6 address fd00:f:f:1::201/64
```

```
!
route-policy pass-all
  pass
end-policy
!
router bgp 65501
 bgp router-id 1.1.1.1
 rpki server 192.168.1.146
  transport tcp port 8282
  refresh-time 15
 !
 address-family ipv4 unicast
  bgp bestpath origin-as allow invalid
 !
 address-family ipv6 unicast
  bgp bestpath origin-as allow invalid
 !
 neighbor 192.168.1.62
  remote-as 65501
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor fd00:f:f:1::62
  remote-as 65501
```

```
  address-family ipv6 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
!
end
```

Appendix A List of Acronyms

AFRINIC	African Network Information Center
APNIC	Asia-Pacific Network Information Center
ARIN	American Registry for Internet Numbers
AS	Autonomous System
BGP	Border Gateway Protocol
BGPsec	Border Gateway Protocol Security
BGP-SRx	BGP Secure Routing Extension
BIO	BGPSEC-IO
CA	Certificate Authority
CPU	Central Processing Unit
eBGP	Exterior Border Gateway Protocol
Gb	Gigabyte(s)
GbE	Gigabit(s) Ethernet
GUI	Graphical User Interface
iBGP	Interior Border Gateway Protocol
IETF	Internet Engineering Task Force
IOS	Internetwork Operating System
IP	Internet Protocol
ISP	Internet Service Provider
IT	Information Technology
JUNOS	Juniper Operating System
LACNIC	Latin America and Caribbean Network Information Center
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
OS	Operating System

RFC	Request for Comments
RIPE NCC	Réseaux IP Européens Network Coordination Centre
RIR	Regional Internet Registry
ROA	Route Origin Authorization
ROV	Route Origin Validation
RPKI	Resource Public Key Infrastructure
RRDP	RPKI Repository Delta Protocol
RTR	Router
SIDR	Secure Inter-Domain Routing
SP	Special Publication
TAL	Trust Anchor Locator
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
VM	Virtual Machine
VRP	Validated ROA Payload
WAN	Wide Area Network

Appendix B References

[NIST BGP-SRx]	*BGP Secure Routing Extension (BGP SRx) Prototype*, National Institute of Standards and Technology, [website]. https://www.nist.gov/services-resources/software/bgp-secure-routing-extension-bgp-srx-prototype
[NIST SP 800-54]	D. R. Kuhn, K. Sriram, and D. Montgomery, *Border Gateway Protocol Security*, NIST SP 800-54, July 2007. http://csrc.nist.gov/publications/nistpubs/800-54/SP800-54.pdf
[NIST SP 800-160]	*Systems Security Engineering: An Integrated Approach to Building Trustworthy Resilient Systems*, NIST SP 800-160 Second Public Draft, National Institute of Standards and Technology, November 2016. http://csrc.nist.gov/publications/drafts/800-160/sp800_160_second-draft.pdf
[RFC 6480]	M. Lepinski and S. Kent, *An Infrastructure to Support Secure Internet Routing*, RFC 6480, February 2012. https://tools.ietf.org/html/rfc6480
[RFC 6482]	M. Lepinski, S. Kent, and D. Kong, *A Profile for Route Origin Authorizations (ROAs)*, RFC 6482, February 2012. https://tools.ietf.org/html/rfc6482
[RFC 6811]	P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein, *BGP Prefix Origin Validation*, RFC 6811, January 2013. https://tools.ietf.org/pdf/rfc6811.pdf
[RFC 7115]	R. Bush, *Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)*, RFC 7115, January 2014. https://tools.ietf.org/html/rfc7115
[RIPE Tools]	*Tools and Resources*, RIPE Network Coordination Centre (NCC), [website]. https://www.ripe.net/manage-ips-and-asns/resource-management/certification/tools-and-resources