LAB: RPKI

Part-1: Installing RPKI Validator

- `#` super user command.
- `$` normal user command.
- Username `apnic` and password `training`.

VM Details

```
[group01.apnictraining.net] [192.168.30.1]
[group02.apnictraining.net] [192.168.30.2]
......
[group10.apnictraining.net] [192.168.30.10]
[group11.apnictraining.net] [192.168.30.11]
......
[group20.apnictraining.net] [192.168.30.20]
[group21.apnictraining.net] [192.168.30.21]
......
[group30.apnictraining.net] [192.168.30.30]
```

Preinstalled packages

To save time, the following essential packages have been preinstalled on the containers:

- `curl`
- `wget`
- `GCC (GNU C toolchain)`
- `rsync`

Lab Setup

For this lab, we will use Routinator from NLnetLabs as the relying party or the RPKI validator.

1. Login to the server (SSH using the username and password given above), where **X** is your group number:

   ```
   ssh apnic@192.168.30.X
   ```

2. Update the repository

   ```
   sudo apt update && sudo apt upgrade
   ```

   Note: Since Routinator is written in `rust`, we will need to first install `rust` using `rustup` (which is a rust installer and version management tool) from the official release channels.

3. Run the following `curl` command which will download a script that downloads `rustup` and installs `rust`

   ```
   curl https://sh.rustup.rs -sSf | sh

   # -f: fail silently (HTTP)
   # -sS: show errors if it fails
   ```

4. Follow the onscreen instructions to install rust:

```
apnic@group01:~$ curl https://sh.rustup.rs -sSf | sh
info: downloading installer

Welcome to Rust!

This will download and install the official compiler for the Rust programming
language, and its package manager, Cargo.

It will add the cargo, rustc, rustup and other commands to Cargo's bin
directory, located at:

  /home/apnic/.cargo/bin

This path will then be added to your PATH environment variable by modifying the
profile file located at:

  /home/apnic/.profile

You can uninstall at any time with rustup self uninstall and these changes will
be reverted.

Current installation options:

   default host triple: x86_64-unknown-linux-gnu
     default toolchain: stable
  modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>
```

5. Make sure to set the PATH environment variable as shown in the onscreen instruction:

```
source $HOME/.cargo/env
```

Note: Before installing Routinator, make sure GCC toolchain is installed:

```
gcc --version
```

```
apnic@group01:~$ gcc --version
gcc (Ubuntu 7.4.0-1ubuntu1~16.04~ppa1) 7.4.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

6. Now we will use `cargo` (the rust package manager) to install Routinator.

```
cargo install routinator
```

```
apnic@group01:~$ cargo install routinator
    Updating crates.io index
  Downloaded routinator v0.3.3
  Downloaded 1 crates (70.4 KB) in 16.67s
  Installing routinator v0.3.3
```

7. Before running Routinator for the first time, we must prepare its working environment (directory for the RPKI cache as well as Trust Anchor Locator - TAL).

```
routinator init
```

Note: Since this is the first time we are using Routinator, it will complain that ARIN's TAL is missing as shown below:

```
apnic@group01:~$ routinator init
Before we can install the ARIN TAL, you must have read
and agree to the ARIN Relying Party Agreement (RPA).
It is available at

https://www.arin.net/resources/manage/rpki/rpa.pdf

If you agree to the RPA, please run the command
again with the --accept-arin-rpa option.
```

8. If we agree to Arin's relying party agreement, reissue the command with the `--accept-arin-rpa` option as shown below:

```
routinator init --accept-arin-rpa
```

```
apnic@group01:~$ routinator init --accept-arin-rpa
Created local repository directory /home/apnic/.rpki-cache/repository
Installed 5 TALs in /home/apnic/.rpki-cache/tals
```

Note: This will create the rpki cache repository directory as well as a download the TALs (from the five RIRs) and save it in the relevant directory.

9. Do a test run with the following command to pull and list the validated ROA payloads (produces a list of ASNs and prefixes). Since it will rsync the whole RPKI repo to the local machine (`/home/apnic/.rpki-cache/repository/`), it will take a while, so dont worry:

   ```
   routinator -v vrps
   ```

   Note: Now you should have all the ROAs from the global RPKI repository on your local validator as a validated cache:

```
rsyncing from rsync://repository.lacnic.net/rpki/.
rsyncing from rsync://rpki.afrinic.net/repository/.
rsyncing from rsync://rpki.apnic.net/repository/.
rsyncing from rsync://rpki.ripe.net/ta/.
rsync://rpki.ripe.net/ta: The RIPE NCC Certification Repository is subject to
Terms and Conditions
rsync://rpki.ripe.net/ta: See http://www.ripe.net/lir-services/ncc/legal/certi
fication/repository-tc
rsync://rpki.ripe.net/ta:
Found valid trust anchor rsync://rpki.ripe.net/ta/ripe-ncc-ta.cer. Processing.
rsyncing from rsync://rpki.ripe.net/repository/.
Found valid trust anchor rsync://rpki.afrinic.net/repository/AfriNIC.cer. Proc
essing.
rsyncing from rsync://rpki.arin.net/repository/.
Found valid trust anchor rsync://rpki.arin.net/repository/arin-rpki-ta.cer. Pr
ocessing.
Found valid trust anchor rsync://rpki.apnic.net/repository/apnic-rpki-root-ian
a-origin.cer. Processing.
rsyncing from rsync://rpki.apnic.net/member_repository/.
Found valid trust anchor rsync://repository.lacnic.net/rpki/lacnic/rta-lacnic-
rpki.cer. Processing.
rsync://rpki.ripe.net/repository: The RIPE NCC Certification Repository is sub
ject to Terms and Conditions
rsync://rpki.ripe.net/repository: See http://www.ripe.net/lir-services/ncc/leg
al/certification/repository-tc
rsync://rpki.ripe.net/repository:
rsyncing from rsync://rpkica.twnic.tw/rpki/.
rsyncing from rsync://rpki-repository.nic.ad.jp/ap/.
rsyncing from rsync://rpki.cnnic.cn/rpki/.
Summary:
afrinic: 338 valid ROAs, 459 VRPs.
lacnic: 2435 valid ROAs, 7042 VRPs.
apnic: 3186 valid ROAs, 21934 VRPs.
ripe: 10780 valid ROAs, 56907 VRPs.
arin: 4964 valid ROAs, 6621 VRPs.
ASN,IP Prefix,Max Length,Trust Anchor
AS43289,2a03:f80:373::/48,48,ripe
AS14464,131.109.128.0/17,17,arin
AS17806,114.130.5.0/24,24,apnic
AS59587,151.232.192.0/21,21,ripe
AS13335,172.68.30.0/24,24,arin
AS6147,190.40.0.0/14,24,lacnic
...
```

NOTE: Now your validator is ready to feed the validated cache to BGP speaking routers through the RTR (RPKI-to-Router) protocol.

Part-2: RTR session

Validator side

Routinator can act as an RTR server, to allow RPKI enabled routers to connect to it and fetch the validated cache (ROA cache).

- IANA has specified a standard port `323` for RTR, which would require running Routinator as a root.
- To run Routinator as a RTR server listening on `192.168.30.X` (where X is your group number) and port `3323`:

```
routinator server --rtr 192.168.30.X:3323 --refresh=900
```

```
apnic@group01:~$ routinator server --rtr 192.168.30.1:3323 --refresh=900
Starting listeners...
```

  - If you dont specify the ***refresh*** time, by default the local repo will be updated and re-validated every 1 hour (as per RFC8210). The example above uses a `15 minutes (900secs)` refresh time

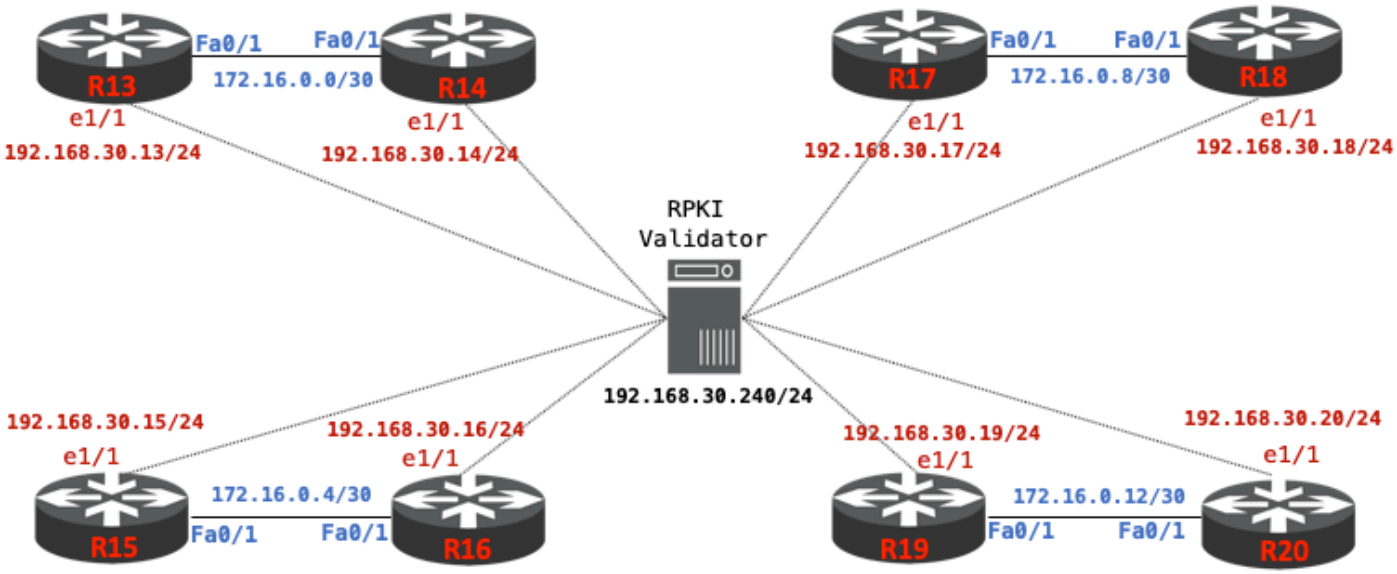
Note: If you have IPv6 address configured on routinator, you can listen on both:

```
routinator server --rtr 192.168.30.X:3323 --rtr [2001:0DB8::X]:3323 --refresh=
900
```

Router Side

Topology

The topology below has 8 routers (`R13, R14, ...R20`), each with a unique ASN (`AS135533 - AS135540`).



Address plan & ROA table

Router	AS#	fa0/1 (to eBGP peers)	e1/1 (to Validator)	ROA/route
R13	135533	172.16.0.1/30	192.168.30.13/24	61.45.248.0/24
R14	135534	172.16.0.2/30	192.168.30.14/24	61.45.249.0/24
R15	135535	172.16.0.5/30	192.168.30.15/24	61.45.250.0/24
R16	135536	172.16.0.6/30	192.168.30.16/24	61.45.251.0/24
R17	135537	172.16.0.9/30	192.168.30.17/24	61.45.252.0/24
R18	135538	172.16.0.10/30	192.168.30.18/24	61.45.253.0/24
R19	135539	172.16.0.13/30	192.168.30.19/24	61.45.254.0/24
R20	135540	172.16.0.14/30	192.168.30.20/24	61.45.255.0/24

Lab Notes

- For this lab, the RPKI Validator (Routinator) has been installed and configured by the instructor as shown in the topology. The validator's IP address is `192.168.30.240`
- To simplify the configuration, the routers will establish eBGP session in pairs as shown below:

```
R13<-->R14
R15<-->R16
R17<-->R18
R19<-->R20
```

- Each router also has a connection to the RPKI validator to allow RTR (**rpki-to-router**) sessions.
- ROAs have already been created for each of the prefixes with corresponding origin AS numbers (`AS135533 - AS135540`) as shown in the table above.

Lab Exercise

1. Telnet to your group's router as shown below:

```
telnet 192.168.30.254 2013 [R13]
telnet 192.168.30.254 2014 [R14]
telnet 192.168.30.254 2015 [R15]
telnet 192.168.30.254 2016 [R16]
telnet 192.168.30.254 2017 [R17]
telnet 192.168.30.254 2018 [R18]
telnet 192.168.30.254 2019 [R19]
telnet 192.168.30.254 2020 [R20]
```

2. If you see the following message during router bootup, enter `no` :

```
Would you like to enter the initial configuration dialog? [yes/no]:
```

3. You also might see the following service configuration messages when the IOS boots:

```
%Error opening tftp://192.168.30.254/network-confg (Timed out)
%Error opening tftp://192.168.30.254/cisconet.cfg (Timed out)
%Error opening tftp://192.168.30.254/router-confg (Timed out)
%Error opening tftp://192.168.30.254/ciscortr.cfg (Timed out)
```

    - Please disable this inbuilt feature and save the config to prevent it from happening during the next boot up:

```
no service config
do wr
```

NOTE: If you need to reload your router, DO NOT issue the reload command (please ask the instructor)!

4. Configure the host name and the interface to the validator (example for R13 below). Refer the address plan table:

```
hostname R13
no logging console
!
interface ethernet1/1
 description link to RPKI-Validator
 ip address 192.168.30.13 255.255.255.0
 no shutdown
```

5. Verify connectivity between the router and the Validator

```
ping 192.168.30.240
```

6. Configure the interface connecting to your eBGP peer (example for **R13** below). Refer the address plan table:

```
interface fa0/1
 description link to R14
 ip address 172.16.0.1 255.255.255.252
 no shutdown
```

7. Verify connectivity to your eBGP peer (talk to your neighbor if there is no reachability). Example for **R19** to check its physical connection to **R20**:

```
ping 172.16.0.14
```

8. Configure eBGP with your neighbor (make sure its the correct neighbor). Example below for **R13**'s eBGP session with **R14**:

```
router bgp 135533
 neighbor 172.16.0.2 remote-as 135534
 !
 address-family ipv4 unicast
  neighbor 172.16.0.2 activate
```

9. Make sure the eBGP session is up with your neighbor

```
sh bgp ipv4 unicast summary
```

    - **Note:** You will not see any prefixes received from your neighbor yet.

10. Announce the correct prefix (based on the address plan table above) to your neighbor. Example below is for **R15**:

```
ip route 61.45.250.0 255.255.255.0 null 0
!
router bgp 135535
 address-family ipv4 unicast
 network 61.45.250.0 mask 255.255.255.0
```

11. Check/Verify routes learned from your neighbor. Example, for **R19** to verify received routes from its neighbor **R20**:

```
sh bgp ipv4 unicast neighbors 172.16.0.14 routes
```

12. Verify the BGP table:

```
sh bgp ipv4 unicast
```

13. Verify the routing table for BGP learned routes

```
sh ip route bgp
```

14. Setup RTR (rpki-to-router) session with the RPKI validator. Example for **R13**:

```
router bgp 135533
  bgp rpki server tcp 192.168.30.240 port 3323 refresh 900
```

NOTE: *Since the router will now pull the validated ROA cache using the RTR protocol from the Validator, it might take a while.*

15. Verify the RTR session with the Validator

```
sh ip bgp rpki servers
```

OR

```
sh bgp ipv4 unicast rpki servers
```

- The output should look like something below:

```
BGP SOVC neighbor is 192.168.30.240/3323 connected to port 3323
Flags 192, Refresh time is 900, Serial number is 0, Session ID is 15578
InQ has 0 messages, OutQ has 0 messages, formatted msg 1
Session IO flags 3, Session flags 4000
Neighbor Statistics:
    Prefixes 39736
    Connection attempts: 1
    Connection failures: 0
    Errors sent: 0
    Errors received: 0
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
……
```

16. Look at all the valid ROAs learned from the Validator

```
sh bgp ipv4 unicast rpki table
```

- Should output a list of ROAs (origin-AS, max-length) like below:

```
65373 BGP sovc network entries using 5752824 bytes of memory
69579 BGP sovc record entries using 1391580 bytes of memory

Network              Maxlen  Origin-AS  Source  Neighbor
1.0.0.0/24           24      13335      0       192.168.30.240/3323
1.1.1.0/24           24      13335      0       192.168.30.240/3323
1.9.0.0/16           24      4788       0       192.168.30.240/3323
1.9.12.0/24          24      65037      0       192.168.30.240/3323
1.9.21.0/24          24      24514      0       192.168.30.240/3323
1.9.23.0/24          24      65120      0       192.168.30.240/3323
1.9.31.0/24          24      65077      0       192.168.30.240/3323
1.9.65.0/24          24      24514      0       192.168.30.240/3323
1.34.0.0/15          24      3462       0       192.168.30.240/3323
1.36.0.0/19          19      4760       0       192.168.30.240/3323
```

17. Now check the BGP table again to see how the routes learned from your neighbors are tagged with the RPKI validation states of **Valid**, **Invalid** or **Not Found**:

```
show bgp ipv4 unicast
```

- Since we have created the ROAs corresponding to the prefixes used in this lab, you should see all of them tagged as valid (**V**). Example below for **R14**:

```
R14#sh bgp ipv4 unicast
BGP table version is 3, local router ID is 192.168.30.14
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
V*>  61.45.248.0/24   172.16.0.1               0             0 135533 i
V*>  61.45.249.0/24   0.0.0.0                  0         32768 i
```

- Also verify the routing table (you should see the valid routes in the routing table)

```
sh ip route bgp
```

18. Let us now try to announce some **Invalid** routes (or hijack someone's routes).

- Go ahead and announce routes (refer the ip address plan) that belong to other groups. In the example below, **R13** in **AS135533** is announcing **R20**'s prefix (**AS135540**):

```
ip route 61.45.255.0 255.255.255.0 null 0
!
router bgp 135533
 address-family ipv4 unicast
  network 61.45.255.0 mask 255.255.255.0
```

- Verify the BGP table on **R14** (your eBGP neighbor).

```
sh bgp ipv4 unicast
```

- You will see that the route `61.45.255.0` learned from its neighbor **R13** has been tagged as **Invalid** (**I**). Discuss within your group why it is Invalid?

```
R14#sh bgp ipv4 unicast
BGP table version is 3, local router ID is 192.168.30.14
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
V*>  61.45.248.0/24   172.16.0.1               0             0 135533 i
V*>  61.45.249.0/24   0.0.0.0                  0         32768 i
I*   61.45.255.0/24   172.16.0.1               0             0 135533 i
```

- Now, look at the routing table:

```
sh ip route bgp
```

OR

```
sh ip route
```

  - You will notice that the Invalid route has **NOT** be been inserted in the routing table.

```
R14#sh ip route bgp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      61.0.0.0/24 is subnetted, 2 subnets
B        61.45.248.0 [20/0] via 172.16.0.1, 00:57:04
```

  - ***NOTE: The default Cisco IOS behaviour is to not include invalid routes for best path selection!***

  - If you dont want to drop invalids with Cisco IOS, you need to explicitly tell BGP to include invalids for best path selection (under respective AFs) as shown below for **R14**:

```
router bgp 135534
 address-family ipv4 unicast
  bgp bestpath prefix-validate allow-invalid
```

- Verify the routing table to see how BGP behaves with the above command:

```
sh ip route bgp
```

- The Invalid route now appears in the routing table of **R14** as shown below:

```
R14#sh ip route bgp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      61.0.0.0/24 is subnetted, 3 subnets
B        61.45.248.0 [20/0] via 172.16.0.1, 01:13:24
B        61.45.255.0 [20/0] via 172.16.0.1, 00:03:15
```

19. Let us have a look at **Not Found** routes - routes for which there are no correspnding ROAs (neither valid or invalid, perhaps not created yet). These make up more than 86% of the global routing table, which indicates many people haven't created ROAs for their prefixes!

    - Let us announce some special use prefixes (RFC5735), for which there should not be any existing ROAs. Example below for **R13** announcing documentation prefix **203.0.113.0/24**

```
ip route 203.0.113.0 255.255.255.0 null 0
!
router bgp 135533
 address-family ipv4 unicast
  network 203.0.113.0 mask 255.255.255.0
```

    - A look at **R14**'s BGP table:

```
R14#sh bgp ipv4 unicast
BGP table version is 5, local router ID is 192.168.30.14
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
V*>  61.45.248.0/24   172.16.0.1               0             0 135533 i
V*>  61.45.249.0/24   0.0.0.0                  0         32768 i
I*>  61.45.255.0/24   172.16.0.1               0             0 135533 i
N*>  203.0.113.0      172.16.0.1               0             0 135533 i
```

- And **R14**'s routing table shows the Not Found routes are included in the best path selection:

```
R14#sh ip route bgp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      61.0.0.0/24 is subnetted, 3 subnets
B        61.45.248.0 [20/0] via 172.16.0.1, 02:02:16
B        61.45.255.0 [20/0] via 172.16.0.1, 00:52:07
B     203.0.113.0/24 [20/0] via 172.16.0.1, 00:03:47
```

20. If we do not want to drop Invalids, we can follow best practice recommendations in **RFC7115** to prefer Valids over Not Found or Invalids, and prefer Not Found over Invalid origins.

    - Define a routing policy that prefers ***Valids > Not Founds > Invalids***

```
route-map ROUTE-VALIDATION permit 10
 match rpki valid
 set local-preference 200
!
route-map ROUTE-VALIDATION permit 20
 match rpki not-found
 set local-preference 100
!
route-map ROUTE-VALIDATION permit 30
 match rpki invalid
 set local-preference 50
```

- Apply the route-map to inbound updates from your neighbor. Example below for **R20**:

```
router bgp 135540
 address-family ipv4 unicast
  neighbor 172.16.0.13 route-map ROUTE-VALIDATION in
```

- Refresh the routes learned from your neighbor (telling them to resend their routes without tearing down the BGP session). Example below for **R14**:

```
clear bgp ipv4 unicast 172.16.0.1 soft in
```

- Now verify the BGP table (example **R14** below) to see the policy in action:
  `sh bgp ipv4 unicast`

```
R14#sh bgp ipv4 unicast
BGP table version is 8, local router ID is 192.168.30.14
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
V*>  61.45.248.0/24   172.16.0.1               0    200      0 135533 i
V*>  61.45.249.0/24   0.0.0.0                  0         32768 i
I*>  61.45.255.0/24   172.16.0.1               0     50      0 135533 i
N*>  203.0.113.0      172.16.0.1               0    100      0 135533 i
```

End of Lab
