



Route Origin Validation Lab

Part-1: Installing RPKI Validator (Routinator)

VM Details

```
[192.168.30.13]
.....
[192.168.30.20]
```

Login Details

- Username `apnic` and password `training`.

Preinstalled packages

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

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

Lab Setup

For this lab, we will use [Routinator](#) from NLnetLabs as the RPKI validator.

1. Login to your server (SSH from the jumphost to your container using the `username` and `password` given above), where `X` is your VM number:

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

2. Update the repository

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

Note: Since Routinator is written in `rust`, 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 --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

```
# -f: fail silently (HTTP)
```

```
# -sS: show errors if it fails
```

4. Follow the onscreen instructions to install rust:

```
apnic@group13:~$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | 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 can be modified with the CARGO_HOME environment variable.

Rustup metadata and toolchains will be installed into the Rustup
home directory, located at:

    /home/apnic/.rustup

This can be modified with the RUSTUP_HOME environment variable.

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
    profile:            default
    modify PATH variable: yes

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

- we will go with the default installation option

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

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

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

```
cargo install routinator
```

```
Updating crates.io index
Downloaded routinator v0.6.4
Downloaded 1 crate (117.5 KB) in 2.28s
Installing routinator v0.6.4
```

◦ **Note:** To update an existing installation, do:

- Rust

```
rustup update
```

- Routinator

```
cargo install -f routinator
```

7. Before running Routinator for the first time, we need to prepare its working environment (directory for the local 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 local rpki cache directory as well as 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 (origin ASN, prefix, max prefix-length).

Note: Since it will rsync/rrdp the whole RPKI repo to the local machine (`/home/apnic/.rpki-cache/repository/`), it will take a while, so dont worry. Instead of printing to standard output, we can write it to a file with the `-o` option:

```
routinator -v vrps -o routinator.csv
```

- Have a look at the validated ROA payload (VRP):

```
more routinator.csv
```

10. **[Optional]** If you have two separate Validators installed (for redundancy/code diversity), compare the validated ROA payload outputs for consistency:

- Example: `diff -u routinator.csv octo.csv`
 - Discuss any differences with your group mates

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 as specified in [RFC8210](#), that periodically fetches RPKI data, verifies/validates it and allows RPKI enabled routers to connect to it to fetch the validated data (ROA cache).

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

```
routinator server --rtr 192.168.30.X:3323 --refresh=300 --detach
```

```
apnic@group13:~$ routinator server --rtr 192.168.30.13:3323 --refresh=300 --detach
```

- If you dont specify the **refresh** time, by default the local repo will be updated and re-validated every 600 seconds (10 minutes). The example above uses a `300secs (5 mins)` 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=300 --detach
```
