An opinionated review of RPKI validators

and the state of their Debian packaging

Marco d'Itri
<md@seeweb.it>
@rfc1036

Seeweb s.r.l.

RIPE 85 - October 26, 2022

Content

seeweb

The software (1)

Validators

- Routinator 3000
- OpenBSD's rpki-client
- ~~RIPE NCC RPKI Validator~~ (discontinued)
- OctoRPKI (not actively developed)
- FORT Validator (no new features until 2023)
- rpki-prover
- ~~Dragon Research Labs RPKI toolkit~~ (not developed since 2018)

seeweb

The software (2)

OctoRPKI and rpki-client do not implement the RPKI-to-router (RTR) protocol themselves, but use an external daemon.

RTR servers

- ~~gortr~~ (abandoned)
- stayrtr

stayrtr is an actively maintained fork of gortr and it looks like it will replace it.

seeweb

Usage of validation software

	October	May
Routinator	79%	69.98%
rpki-client	8%	19.30%
RIPE NCC Validator	4%	4.37%
OctoRPKI	6%	3.53%
FORT Validator	3%	3.23%
rpki-prover	0%	0.52%

This is dangerously close to becoming a *software monoculture*.

This data was gathered by Job Snijders by counting the unique IPs accessing a RRDP web server.

seeweb

Routinator

Pros

- Actively developed, support contracts available.
- Well documented.

Cons

- Impossible to package by distributions.
- Too high adoption causes a lack of software diversity.

Developed in Rust by NLnet Labs.

seeweb

rpki-client

Pros

- Actively developed by network operators, support contracts available.
- Simple and essential.
- Separation of privileges in multiple processes.
- Implements many new features.

Cons

- Needs a third party RTR daemon.

Developed in C by the OpenBSD project.

seeweb

RIPE NCC Validator

Pros

- Nothing else was available at the time?

Cons

- Written in Java.
- RIPE NCC stopped development.
- End of support in June 2021: **nobody should use it anymore!**

Developed in Java by RIPE NCC.

seeweb

OctoRPKI

Pros

- Simple and essential.

Cons

- Not developed anymore except for security fixes since the original author left Cloudflare.
- Needs a third party RTR daemon.

Developed in Go by Cloudflare.

seeweb

FORT Validator

Pros

- Used to be actively developed.
- Well documented.
- Good middle ground of features and complexity.

Cons

- Currently in bug-fix only mode, development will resume in 2023.

Developed in C by LACNIC and NIC.MX.

seeweb

rpki-prover

Pros

- Software diversity is good.

Cons

- Niche programming language.
- Very low adoption.

Developed in Haskell by Mikhail Puzanov.

Should I package it?

seeweb

My suggestions

Use two of:

- Routinator
- FORT Validator
- rpki-client + stayrtr

They are all good and have different tradeoffs.

Using software packaged by a Linux distribution significantly reduces the system administration effort and allows to adopt diverse implementations.

Software diversity is important and needs to be encouraged!

seeweb

Features

	BGPSec	ASPA	RSC	signed TALs
Routinator	✓			
rpki-client	✓	✓	✓	✓
OctoRPKI				
FORT Validator				
rpki-prover			✓	

seeweb

Content

seeweb

Why use packaged software

The great debate: packages from distributions[1] or the developers?

Why use distribution packages?

- Integration with the OS and high attention to details.
- Ready to use after the installation.
- Automatic security updates[2].
- Maintained by system administrators, not software developers.

Why use vendor packages?

- Freshness.

[1] Full disclosure: I develop a Linux distribution (Debian).

[2] Job estimated that over 70% of the clients currently in use are insecure.

seeweb

Debian for network operators

Debian GNU/Linux is the one stop shop for all your RPKI validation needs.

My goals

- Packages with sane defaults which just work after being installed.
- Common management of TALs in the `rpki-trust-anchors` package.
- State of the art security with systemd sandboxing.

Issues

- The RPKI ecosystem is still young and fast moving for a stable distribution.
- Routinator cannot be packaged.

seeweb

The issue with Routinator

The Rust development ecosystem is broken and hostile to distributions

- APIs are not stable (and there is no dynamic linking).
- Hence it is common for Rust software to depend on specific versions of libraries.
- General *vendoring* of dependencies is not acceptable to the Debian security team.
- Maintaining multiple versions of libraries in the distribution is too much time consuming (and not appreciated either...).
- Different Rust programs depend on different versions of the same library.
- **There is no practical way to package complex Rust projects.**

The Routinator developers publish a Debian package which is good enough, but it does not use `rpki-trust-anchors`.

The state of Debian RPKI packages

Package	Debian 11	Debian testing	Ubuntu 22.04
routinator	✗	✗	✗
rpki-client	✓	✓	✓(7.6)
cfrpki	✓	✓	✓
fort-validator	✓	✓	✓
gortr	✓	✓	✓
stayrtr	✗	✓	✓
rpki-trust-anchors	✓	✓	✓
OpenBGPD (bonus!)	✗	✓	(old)

`stayrtr` is not in Debian 11, but `gortr` still works fine.

Ubuntu 22.04 LTS is good right now but the packages will probably not be updated over its life.

seeweb

Backports to Debian/stable

Backported packages of `rpki-client` will be maintained in the official Debian backports archive until the release of Debian 12.

```
echo 'deb http://deb.debian.org/debian bullseye-backports main' \
  > /etc/apt/sources.list.d/bullseye-backports.list
apt update
apt install rpki-client/bullseye-backports
```

I plan to backport other RPKI-related packages too if and when it will be needed.

seeweb

Any questions?

https://www.linux.it/~md/text/rpki-validators-ripe85.pdf

(Google ... Marco d'Itri ... I feel lucky)

seeweb