Emulation and Operation Monitoring (EOM) Tool

Design Document

Parsons Corp.

Aug, 2015

1 Purpose of the Tool

The purpose of this tool is to provide an emulation environment in which an ISP (or other BGP AS) can analyze the impact of planned deployment of RPKI validation in local conditions, without impacting their routing operations. The EOM Tool is expected to assist the operator in scenarios such as the following.

- Assessing if the best route selection changes when validation is enabled.
- Assessing the manner in which the best path changes when different local pref/weight settings are used.
- Combining data from multiple routers in different ASNs (an ISP can have more than one AS) to check for any routing-level inconsistencies.
- Assessing potential problems associated with validating prefixes in a multihoming environment.
- Assessing whether an ISP's reliance on multiple rpki-rtr manager instances could result in routing-level inconsistencies.
- Testing custom prefix assignment and certification conditions for simulation of failures and other special scenarios - e.g. resource transfers.

2 Tool Components

The block diagram above highlights the different sub-components within the EOM tool. At a high level the EOM tool will fetch RPKI information from one or more RPKI caches, poll RIB related information from routers, analyze the different pieces of data retrived from these data sources against various parameters, and display its results through a web-based user interface.

The following sub-sections describe the functionality associated with each EOM tool sub-component.

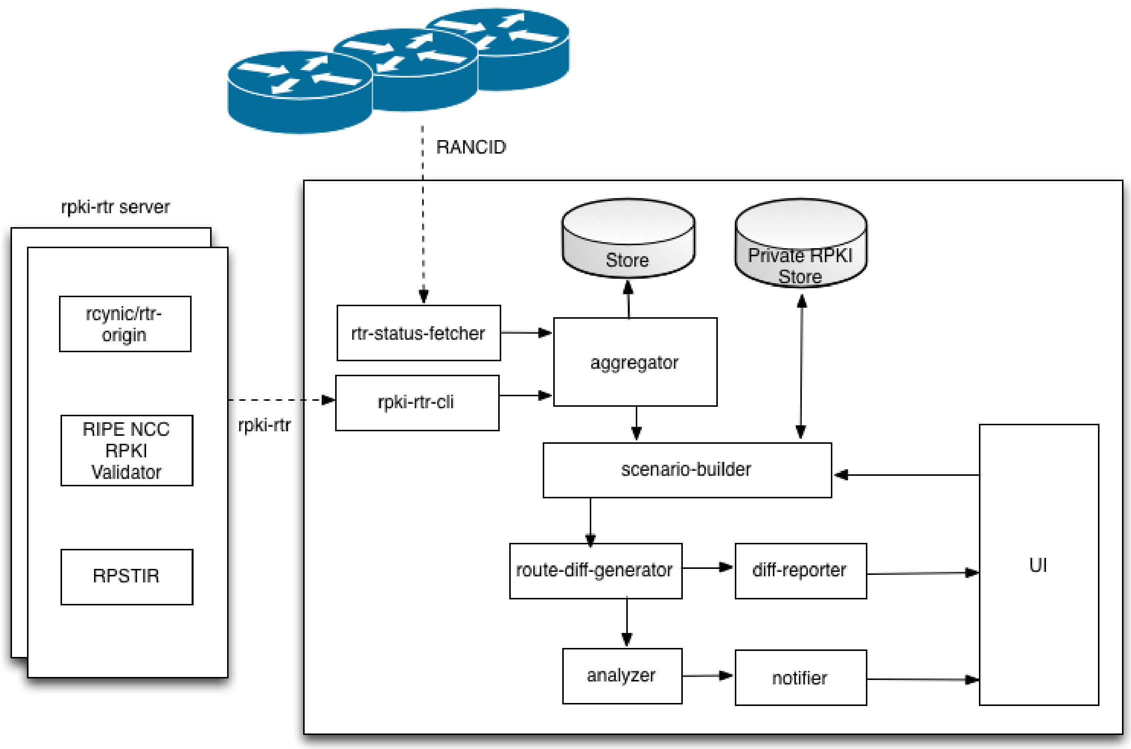

Figure 1: Tool Component Block Diagram

2.1 rpki-rtr-cli

This component will provide the client interface to the rpki-rtr protocol. It will most likely build upon existing rpki-rtr implementations but instead of communicating with a router this module will interface with an aggregator module that maintains a historical record of validated RPKI information.

The rpki-rtr-cli module will have the capability to communicate with multiple rpki-rtr manager instances, inter-operating with multiple rpki-rtr server implementations, in order to be able to mimic cases where an ISP relies on multiple rpki-rtr manager instances for serving validated RPKI information to its various ASBR routers.

2.2 rtr-status-fetcher

This module will be responsible for fetching various pieces of status information from a router, including its routing table and next hop neighbors. Data from a router will be fetched through RANCID, an open-source tool that simplifies the automation of routine operations on routers through expect scripts. The EOM tool may be required to analyze the data for multiple routers, so the rtr-status-fetcher will also save all fetched data into a local store through the aggregator module.

2.3 aggregator/store

This module will be responsible for storing and retrieving validated rpki route objects associated with various rpki-rtr manager instances and router status information for the different routers queried. New data will be fed by the rpki-rtr-cli and rtr-status-fetcher modules and existing data will be queried by other modules that wish to analyze such data under various operational scenarios.

2.4 scenario builder

The scenario-builder module will be responsible for providing the central funcionality for enabling a network operator to configure and develop various types of operational scenarios and router policies to study the impact of RPKI validation on their network operations.

This module will interface with the UI module to gather user input on the types of analyses that are to be performed. It will access previously saved router and RPKI cache data through the aggregator module and prepare the data for subsequent analysis by other modules.

In order to build scenarious that include misconfigured or particular forms of RPKI validation data, the scenario-builder will also interface with a private RPKI store that will allow the user to build test ROAs on the fly to simulate various prefix assignment and certification conditions.

2.5 route-diff-generator

This module will provide the first level of analysis over the different streams of data that are made available by the scenario-builder. The primary function of this module will be to detect and flag differences in the routing state for different operational scenarios.

2.6 diff-reporter

The diff-reporter module will take the raw results generated by the route-diff-generator module and will transform that data to a form that is useful for user consumption. Differences will initially be represented as simple text but will later be color coded and marked up to clearly highlight any differences.

2.7 analyzer

The analyzer module will operate on data generated by the route-diff-generator to make inferences about potential routing level inconsistencies that may be useful to flag to the user.

2.8 notifier

The notifier module will be responsible for processing the results from the analyzer module in order to present those results to the user in an intuitive manner in order to help them develop additional test scenarios in an iterative manner. In the future, when the EOM tool is used as a monitoring tool, this module will also provide user notification capabilities to warn and alert the user of existing and impending problems.

2.9 UI

The User Interface module will provide the engine for displaying the various pieces of data generated by the data-reporter and notifier modules. This module will also provide the configuration interface for the users, to enable them to specify the location and parameters associated with the different routers and RPKI stores, and to enable them to define the parameters associated with their different scenarios of interest.

3 Development Phases Overview

This section outlines the likely progression of the various tool capabilities.

3.1 Phase 1

In the initial phase we will add basic support for the rpki-rtr-cli, rtr-status-fetcher, and route-diff-generator modules so that the tool provides the ability to analyze routing information extracted from a border router and produce a one time text report of route validity.

The aggregator, analyzer, notifier and scenario-builder modules will be stubbed out and will provide only basic API compatibiity.

3.2 Phase 2

In the second phase we will extend rtr-status-fetcher to periodically poll multiple routers for their RIB information, and will also add initial capability to the aggregator, route-diff-generator and diff-reporter modules to produce enhanced reporting capabilities through a web-based front-end.

We will also add initial capability to the scenario builder to help the operator assess the effects of RPKI validation under different router policy (e.g. local-pref) settings.

3.3 Phase 3

In this phase we will enhance the EOM tool to provide post-deployment monitoring support in addition to supporting pre-deployment planning.

The scenario-builder component will be enhanced to support the inclusion of custom RPKI data fed through a private RPKI store, which will be anchored to a private Trust Anchor.

The analyzer component will be enhanced to automate the analysis of various pieces of data in order to provide the operator with the capability to monitor the consistency of their RPKI and routing data on an ongoing basis.

The nature of analysis changes once RPKI validation has been enabled within an operational environment. Here the EOM tool must make a determination about how the absence (rather than the presence) of validation is likely to impact best path selection. Since routers may not have the capability to display their RPKI state even when they actually use RPKI information in their best path selection, inferring the best path in the absence of validation may be non-trivial. The specifics of this analysis will need to be revisted in due course.

4 Synergy with other Efforts

A number of implementations exist for the server and client components of the rpki-rtr protocol. The client component of the rpki-rtr protocol is usually integrated into router implementations. However, currently, there is no tool that allows the user to compare the data from routers juxtaposed with RPKI validation data in order to study the effects of RPKI validation on operational routing. The EOM tool fills this gap. However, where possible, the EOM tool will interoperate with and build upon existing implementations of the rpki-rtr protocol in order to minimize duplication of effort.

The EOM tool will also build upon other route collection efforts. For example, the tool is envisioned to interact with multiple rpki-rtr servers, some of which may be external to the ISP/network operation where the EOM analysis is being performed. Likewise, the EOM tool will also fetch data from various public looking glasses in order to provide external validity of the routing information being analyzed.

Finally, the EOM tool components will be open sourced in order to encourage collaborative participation by the operator/user community and to widen the use of the tool as a deployment aide.