Updates from the Internet Backbone: An RPKI/RTR Router Implementation, Measurements, and Analysis

Matthias Wählisch
Freie Universität Berlin
Berlin, Germany
waehlisch@ieee.org

Fabian Holler, Thomas C. Schmidt
HAW Hamburg
Hamburg, Germany
mail@fholler.de, t.schmidt@ieee.org

Jochen H. Schiller
Freie Universität Berlin
Berlin, Germany
jochen.schiller@fu-berlin.de

Abstract

A fundamental change in the Internet backbone routing started in January 2011: The Resource Public Key Infrastructure (RPKI) has officially been deployed by the Regional Internet Registries. It leverages the validation of BGP prefix updates based on cryptographically verified data and may lead to secure inter-domain routing at last. In this talk, we present RTRlib, a highly efficient reference C implementation of the RPKI router part. We deploy RTRlib and conduct a long-term measurement using live BGP streams to evaluate the current impact of RPKI-based prefix origin validation on BGP routers. We observe that most of the invalid prefixes are most likely the result of misconfiguration. RTRlib is the only openly available tool for monitoring RPKI validation activities in real-time. We measure a relatively small overhead of origin validation on commodity hardware (5% more RAM than required for full BGP table support, 0.41% load in case of $\approx$ 92,000 prefix updates per minute).

1. Introduction

The Internet backbone is based on the Border Gateway Protocol (BGP). BGP announces IP prefixes to enable inter-domain routing between so called Autonomous Systems (ASes). One major problem of BGP is its lack of verifiable information exchange. Several prominent incidents highlighted the consequences [6, 1]: An AS incorrectly claims to own an IP prefix and thereby redirects traffic, which not only may lead to traffic interruption, but can be used to intercept and tap data streams. Only recently, countermeasures have been deployed in the form of the Resource Public Key Infrastructure (RPKI) [4] and related protocols. A successfully deployed RPKI origin validation would have immediately disclosed the prefix hijacks referenced above—a rigorous route rejection of the invalid updates would have prevented the incidents entirely.

The RPKI stores cryptographically provable mappings of IP prefixes to ASes that are legitimate to originate these prefixes. The corresponding attestation objects are called *Route Origin Authorization* (ROA). To prevent BGP routers from cryptographically load, external cache servers verifies ROAs and transmit only valid ROA data to the BGP router using the RPKI/RTR protocol [2]. In combination with an origin validation scheme for IP prefixes [5], a router is able to verify the correctness the announced origin AS. According to the validation outcome, a BGP prefix update may be valid, invalid, or not found in the RPKI.

In this talk, we analyze the impact of prefix origin validation on BGP routers and the potential consequences for the current BGP-based route propagation. We introduce the RTRlib, a real-time compliant, highly efficient implementation to secure inter-domain routing at BGP peers. This open-source software is a reference implementation of the latest IETF protocol standards to perform prefix origin validation, and written in C. It features a flexible architecture and can be used to extend existing BGP daemons at real routers but also to implement new monitoring and analysis tools in the context of RPKI/BGP research.

Based on live BGP update streams representing more than 100 peering neighbors, we present a long-term measurement highlighting two months which verifies 420 million IP prefix updates against the available ROA data. We observed that most of the invalid prefix announcements are most likely due to misconfiguration of the attestation objects. Our observations do not suggest to apply strict rejection of invalid prefix updates at the moment. Furthermore, we found single events initiating a significant amount of validations.

We extract the key lessons learned from the data observed during our measurement period and derive advice for ISPs on future operational use of the RPKI. We systematically explore the overhead of prefix origin validation at commodity router hardware. Enabling RPKI will require $\approx$ 5% more RAM compared to the storage of the global BGP routing table. The CPU load depends insignificantly on the RPKI-deployment state.

The remainder of this abstract is structured as follows.

In § 2, we present the architecture of the RTRlib as well as a performance overview and our insights from live BGP updates. We conclude in § 3.

2. RTRlib – A C Library for RPKI/RTR Router Support

To extend routing by an RPKI-based prefix origin verification, the RPKI/RTR protocol needs to be implemented on routers. RTRlib is the first full-fledged open-source C implementation that is suitable for testing purposes as well as production use. We assembled the required functions as an external independent library, which simplifies code reuse. Existing BGP daemons can be extended by simply integrating the RTRlib or parts of it. The same code base may also be used to build tools for researchers or ISPs (e.g., to monitor the RPKI). The software follows the design principles of broad system integration, interoperability, extensibility, and efficiency. RTRlib is licensed under GNU LGPL and available at `http://rpki.realmv6.org`.

Architecture The software architecture includes different layers to simplify the extension or an exchange of individual parts. The lowest layer of the architecture is built by the *transport sockets*. They allow for the implementation of different transport channels that provide a common interface to exchange PDUs with the cache (i.e., the RPKI/RTR server). The current version of the library supports unprotected TCP and SSH. On top of the transport layer, the *RTR socket* uses a transport socket for RTR-specific data exchange with the RPKI/RTR server. The RTR socket implements the RPKI/RTR protocol, i.e., fetches validation records and stores them in a prefix table data structure.

The *prefix validation table* stores validated prefix origin data. This abstract data structure provides a common interface to add and delete entries as well as to verify a specific prefix. Its implementation is crucial as the data structure stores all prefixes received from the cache servers (i.e., low memory overhead required) and is responsible to perform prefix lookup for the BGP updates (i.e., find validated IP prefixes very fast). Our library implements a Longest Prefix First Search Tree (LPFST) [8], but can be extended to other data structures. Internally, the RTRlib uses two separate prefix validation tables, one for IPv4 records and one for IPv6 records. This makes tree operations (insert, delete, find) more efficient as the height per tree is lower in contrast to a combined IPv4/v6 tree.

On top of the modular architecture, the *RTR connection manager* maintains the connection to multiple RTR servers. This includes failover mechanisms. It represents the main interface for users of the library.

Performance Overview We measure the runtime performance and the scaling behaviour based on (1) experiments and (2) live BGP streams to quantify the overhead introduced by RPKI-based prefix origin validation at BGP routers. The measurement node consists of commodity hardware with a dual-core AMD Opteron 280 processor (2.4 GHz) and 8 GB RAM. The operating system was Linux, kernel 2.6.32-33.

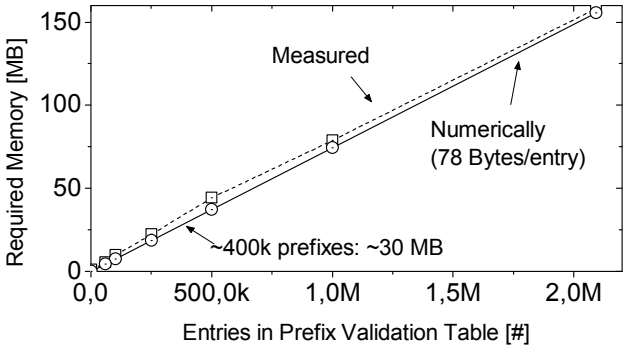

Figure 1. Memory consumption of RTRlib

The *memory consumption* of the library mainly depends on the number of prefixes inserted into the prefix validation table. Considering a 64 bit architecture with 8 bytes per pointer, a single record within the prefix validation table consumes 78 bytes in our implementation of the LPFST. To measure the memory required on a real system, we added randomly generated prefixes to the prefix validation table. The overall memory consumption scales linearly for different table sizes as expected (cf., Fig. 1). ROAs for all $\approx$ 400,000 active IP prefixes included in current BGP routing tables would result in additional $\approx$ 30 MB of RAM for an RPKI/RTR-enabled router. Thus a full RPKI validation table would lead to a 5% increase of RAM [3].

The *processing overhead* of RPKI/RTR on the router is dominated by the complexity that results from update and lookup operations on the data structure holding the valid ROA information. Update operations on the prefix validation table are triggered by new, modified, or deleted ROAs, whereas lookups follow BGP updates. The asymptotical complexity of the LPFST is $O(n \cdot \log(n))$ with n entries. 1 million entries can be imported in $\approx$ 4 seconds, which allows for a fast creation of the prefix validation table and prompt start of origin verification after a reboot. We also analyzed the CPU overhead depending on different potential states of RPKI deployment by randomly generating 100,000 different ROA data. The performance evaluation is based on a predefined ratio of validation state of 0%, 25%, 50%, 75%, and 100%. For each combination of all possbile validation outcomes (e.g., 25%valid, 50% invalid, and 25% not found), we measured dependency on the input set below one clock tick. For current deployment, this is negligible. However, we argue that even a slight dependency may be misused by an attacer, e.g., to perform a complexity attack.

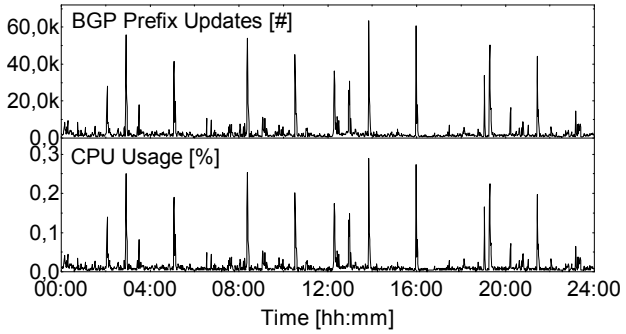

Figure 2. Characteristic CPU load Jan. 5, 2012

RPKI in the Wild To evaluate the behaviour of an RPKI/RTR-enabled router under real conditions, we extend our setup. The measurement node maintains a connection to (1) an officially deployed RTR cache server and (2) BGPmon. BGPmon provides a live BGP stream of nine direct and three indirect peerings. The indirect peering includes updates from more than 100 peers. We measure the CPU load and record the IP prefix, mask length, and AS path as well as the validated ROA data if the prefix is included in the RPKI. We consider the months January and May 2012.

On average we received ≈ 6 million prefix updates per day. The CPU load corresponds to the number of prefix validations (i.e., the BGP update rate). Figure 2 visualizes both measurements per minute for January 5, 2012. All other days show the same qualitative behaviour. During the measurement period, we observed a maximum of 92,549 prefix announcements per minute and a maximum CPU load of 0.41%. The average CPU load per day was 0.02% with a standard deviation of 0.04%.

Most of the announced IP prefixes are stable in the sense that they remain visible over a longer period of time. They are advertised continuously. To prevent prefix hijacking, each RPKI-enabled BGP router must process any single prefix advertisement and verify the origin AS against the currently valid ROA data. However, not all valid prefixes initiate a change in the Routing Information Base (RIB). In particular, prefix updates that are already part of the RIB need not to be evaluated again if received in a short period of time. Caching might be used to optimize access time, for example. From this perspective we quantify the difference between the information a router *sees* in the updates and the new information a router *learns*. On average, the number of successful prefix origin validations is lager than the invalid prefixes. In contrast to January, the amount of valids increased and the number of invalids decreased in May. We observed 71,619 valid prefix updates (and 3598 unique prefixes) on average in January, and a mean of 131,764 prefix updates (and 5047 unique prefixes) in May. Our measurements indicate that most storms of valid/invalid prefixes are due to a high frequency of BGP updates and not based on changes of the prefix/RPKI data.

3 Conclusion and Outlook

This talk presented a first practical exploration of the Resource Public Key Infrastructure (RPKI) recently released by the IETF. In an evolutionary approach, RPKI allows for authenticating prefix-to-AS mappings in BGP route advertisements without altering the Internet backbone routing. We introduced the first full-fledged RPKI/RTR router implementation in C that is available for public download. Our performance analysis revealed its readiness not only for research and monitoring, but also for production-type services. The second part of our work was dedicated to a long-term measurement and analysis of real-world Route Origin Authorization (ROA) management and the validation of prefixes in real-time BGP streams. We monitored an emerging deployment of operators and increased quality of RPKI data.

Currently we work on establishing an online monitoring service that displays the status of RPKI prefix validation in near real-time. We will extend our analysis about the vulnerability of an RPKI-enabled router and conduct detailed study of the identification of prefix hijacks [7].

Acknowledgements We would like to thank the SIDR community and Olaf Maennel for very valuable discussions. This work is supported by the German BMBF within the project Peeroskop.

References

[1] M. A. Brown. Pakistan hijacks YouTube – Renesys Blog, February 2008.

[2] R. Bush and R. Austein. The RPKI/Router Protocol. Internet-Draft – work in progress 26, IETF, February 2012.

[3] Cisco. BGP: Frequently Asked Questions. `http://www.cisco.com/image/gif/paws/5816/bgpfaq_5816.pdf`, 2012.

[4] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480, IETF, February 2012.

[5] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. BGP Prefix Origin Validation. Internet-Draft – work in progress 10, IETF, October 2012.

[6] D. M. Slane, C. Bartholomew, et al. 2010 Report to Congress. Annual report, U.S.–China Economic and Security Review Commission, November 2010.

[7] M. Wählisch, O. Maennel, and T. C. Schmidt. Towards Detecting BGP Route Hijacking using the RPKI. In *Proc. of ACM SIGCOMM, Poster Session*, pages 103–104, New York, August 2012. ACM.

[8] L.-C. Wuu, T.-J. Liu, and K.-M. Chen. A longest prefix first search tree for IP lookup. *Computer Networks*, 51(12):3354–3367, August 2007.