Path Plausibility Algorithms in GoBGP

Nils Höger*, Nils Rodday*†, Oliver Borchert‡, Gabi Dreo Rodosek*

*Research Institute CODE, Universität der Bundeswehr München,
†University of Twente ‡National Institute of Standards and Technology

***Abstract*—BGP is known to be inherently insecure. Many solutions have been proposed, with the RPKI becoming operational in 2011. The RPKI only provides origin validation. This leaves path manipulation attacks and route leaks unsolved. In this demo session, we present our integration of two path plausibility algorithms, namely ASPA and AS-Cones, into the GoBGP routing daemon. We present advantages and drawbacks of both approaches and extend the NIST BGP-SRx software suite with the first implementation of AS-Cones.**

***Index Terms*—BGP, ASPA, AS-Cones, Path Plausibility**

I. Introduction

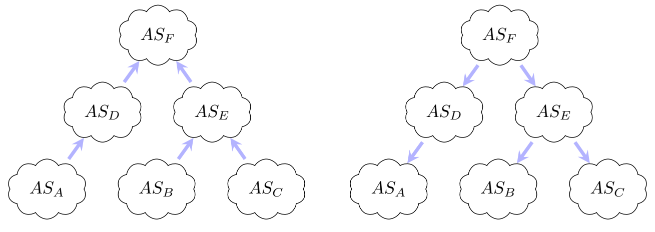
The Border Gateway Protocol (BGP) enables the exchange of reachability information within the inter-domain routing infrastructure. Unfortunately, it was never designed to be run in an untrusted environment and therefore assumes proper behavior of every participant. In an infrastructure with currently 74,110 different Autonomous Systems (ASes) spread across the globe, trust cannot be taken for granted. Several attack vectors are present in the Internet routing infrastructure such as BGP prefix hijacks [1], path manipulation attacks [2], and route leaks [3] [4].

Many security solutions have been proposed throughout the years (*e.g.,* psBGP, S-BGP, soBGP) that did not succeed to become standardized. To solve the security-related shortcomings of BGP, the Internet Engineering Task Force (IETF) created the Secure Inter-Domain Routing (SIDR) working group which standardized two mechanisms:

Prefix Origin Validation. The Resource Public Key Infrastructure (RPKI) [5] was operationally deployed in 2011 as the first cryptographically secured countermeasure against BGP prefix hijacking. It allows prefix owners to create and publish a Route Origin Authorization (ROA) object which contains information as to which AS is authorized to announce the given prefix. Other ASes can fetch the ROA objects and make routing decisions based on the outcome of RPKI Route Origin Validation (ROV) of received BGP updates. RPKI ROV only protects against BGP prefix hijacks where the origin in the BGP announcement does not match the legitimate AS specified in a ROA object. Path manipulation attacks and route leaks cannot be detected using RPKI ROV.

Path Validation. Border Gateway Protocol Security (BGPsec) [6] aims at cryptographically securing the entire AS path within a BGP announcement. However, it has some drawbacks that currently inhibit deployment on a wider scale. First, it does not support partial deployment and thus is only functional along contiguous ASes that all support BGPsec. A single non-BGPSec capable router in-between breaks the cryptographic chain and downgrades the announcement to regular BGP. Second, BGPsec is, in contrast to RPKI-ROV, designed to digitally sign and validate signatures in-band in the BGP protocol. These cryptographic operations need to be executed on the router, which comes with significant costs in performance. Third, BGPsec does not permit BGP update packing, which will result in routing tables growth when deployed. Lastly, it does not protect against route leaks.

Path Plausibility. In this work, we focus on *path plausibility* algorithms, in particular Autonomous System Provider Authorization (ASPA) [7] and AS-Cones [8]. The effectiveness of path plausibility algorithms is currently a hot topic within the IETF's Secure Inter-Domain Routing Operations (SIDROPS) working group [9], the successor of SIDR. Path plausibility algorithms are out-of-band mechanisms that are used to check the plausibility of an AS path. They cannot cryptographically prove that the BGP announcement traversed the path it claims but only state the possibility of whether a BGP announcement could have potentially traversed through the advertised AS path. Figure 1 illustrates the proposed RPKI object creation process for each algorithm. ASPA works in a bottom-up fashion where an AS publishes an ASPA object specifying its providers. AS-Cones works in a top-down fashion where each AS publishes an AS-Cones object that specifies all customers. Since there exist many more customer ASes compared to provider ASes, the number of objects that have to be created for ASPA is much greater than the number of objects required for the AS-Cones approach. Next to object creation ASes will have to implement the filtering algorithm based on the



(a) ASPA (b) AS-Cones

Figure 1: Comparison of ASPA and AS-Cones object creation: To cover the whole graph, ASPA requires the creation of five objects, while AS-Cones requires the creation of three objects.

particular draft [7], [8]. Even though these algorithms do not provide the path protection that BGPsec provides, they are designed to operate in partial deployments and do not add a significant additional load on routers.

Contributions. In this work, we propose a BGP router implementation called GoBGPSRx which supports both, ASPA and AS-Cones validation. In detail, we make the following contributions:

1) We extend the GoBGPsec implementation to support ASPA and AS-Cones validation.
2) We provide an implementation for AS-Cones within the National Institute of Standards and Technology (NIST) BGP Secure Routing Extension (SRx) software suite.
3) We create a container-based solution to easily integrate our implementation into other projects and design a testbed for evaluation.
4) We publicly release all source code and documentation of our extensions [10].

II. NIST BGP-SRx Software Suite

Our work relies heavily on the NIST BGP SRx software suite [11] which offers multiple components:

SRx-Server. The SRx-Server implements several standards: RPKI [5], BGPSec [6], and ASPA [7] (draft version 9). We extend the SRx-Server implementation with the AS-Cones [8] algorithm. It receives validation requests from a BGP routing daemon and replies with the validation results. RPKI objects, such as ROAs, ASPA, and AS-Cones objects are collected via the RPKI to Router (RTR) protocol from the RPKI test harness. We extend the RTR protocol [12] with an AS-Cones Protocol Data Unit (PDU).

BGP routing daemon. The BGP routing daemon connects to the SRx-Server to perform validation requests. The NIST BGP-SRx's default routing daemon is based on Quagga. In addition, the framework provides GoBGPsec, which is based on the GoBGP routing daemon. We extend the GoBGPsec daemon with an additional GoSRx-Proxy Application Programming Interface (API) for ASPA and AS-Cones validation.

RPKI Cache Test Harness. The RPKI Cache Test Harness provides a database for validated RPKI information such as Validated ROA Payload (VRP), Validated ASPA Payload (VAP), and BGPSec keys. We extend the RPKI Cache Test Harness to also support validated AS-Cones payload.

BGPsec-IO. The BGPsec-IO (BIO) is a BGP and BGPsec traffic generator that is capable of generating multi hop BGP paths and fully signed multi hop BGPsec paths. It is used for the evaluation of BGP and BGPsec router implementations.

III. Methodology & Implementation

The NIST BGP-SRx provides two methods for validation: local and remote. Local validation is only available for BGPsec using the SRx Crypto API (SCA) directly from within the daemon. Remote validation outsources the validation to the SRx-Server. The SRx Proxy facilitates communication between a BGP daemon and the SRx-Server. It implements the SRx

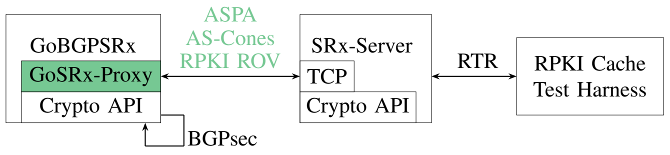


Figure 2: GoBGPSRx architecture: The highlighted part shows our GoSRx Proxy implementation, which connects via a TCP socket to the SRx-Server.

Proxy Protocol within the SRx Proxy API [13], which hides all complexities of TCP protocol communication with the SRx-Server. Moreover, it offers full support for RPKI, BGPsec, ASPA, and (in our extended version) AS-Cones validation. Another method to communicate with the SRx-Server is to directly implement the SRx Proxy Protocol without using the SRx Proxy API. On the one hand, it is much more complex as asynchronous processes have to be dealt with. On the other hand, it offers greater flexibility as it allows the use of other programming languages. Building a Go-based SRx-Proxy allows direct integration into the GoBGP daemon.

Choice of interface. The NIST BGP-SRx framework extends the GoBGP daemon with the capability to perform RPKI and BGPsec requests towards the SRx-Server. The extended version uses the SCA to enable BGPsec features for the existing BGP daemon. It is now called GoBGPsec [14]. Our work builds upon the GoBGPsec daemon and adds ASPA and AS-Cones validation features. We call the resulting daemon GoBGPSRx [10]. We added an additional API that uses socket-based communication between the two entities. Figure 2 illustrates the additional interface we created.

Implementation. We introduce two new classes: RPKI-manager and Go-Proxy. During startup, the routing daemon creates a new RPKI-manager instance that is responsible for ASPA and AS-Cones validation. The manager spawns a new GoSRx Proxy instance responsible for the session handling between the SRx-Server and GoBGPSRx. The GoSRx Proxy connects to the SRx-Server via the SRx Proxy Protocol [13] implemented in the Go programming language. Depending on the message type contained in the replies received from the SRx-Server, the GoSRx Proxy invokes the respective callback functions within the RPKI-manager to process the received data. If the router receives an update message from a BGP peer, it forwards it to the RPKI-manager. The RPKI-manager stores the message in a separate data structure and extracts the mandatory information to create a validation message. The validation request can either be of type ASPA or AS-Cones. The message contains the AS path list, the propagated prefix, and a default result, among other fields. The RPKI-manager forwards this message to the GoSRx Proxy, which then sends the validation request to the SRx-Server. The SRx-Server sends the validation result back to the GoSRx Proxy [11], which forwards the data to a callback function of the RPKI-manager. The manager ignores the stored BGP message if the validation result is *invalid* or *unknown*. If the result is *valid*, it passes

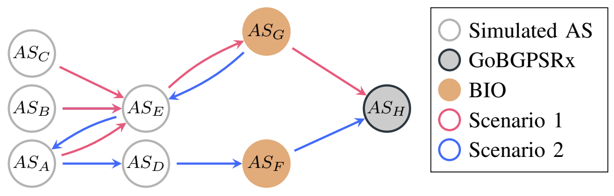


Figure 3: EvaluationTopology

the BGP message back to the routing daemon, which further processes the data.

We also created a Docker image to properly test the implementation in a testbed. This image extends the one from the NIST BGP-SRx framework and is also based on centos. It contains all components of the NIST-BGP-SRx framework, including our router implementation.

IV. Proof of Concept

We evaluate our router implementation using the demo topology, which is part of the NIST BGP-SRx software suite for ASPA evaluation. Figure 3 displays the topology of our testbed. The directed lines represent the direction of BGP announcements.

In this topology, we replace AS_H with our GoBGPSRx implementation and simulate incoming BGP traffic from the neighbors AS_G (Scenario 1) and AS_F (Scenario 2) using the BIO traffic generator. We run several use cases to test our implementation: regular BGP messages, BGP prefix hijacks combined with path-shortening, and route leaks. Our implementation can recognize valid BGP messages and can detect artificially crafted attacks through validation using the SRx-Server. After successful validation, only valid BGP messages are entered into the local Route Information Base (RIB). All other messages are discarded. Thus, we are certain that our implementation performs validation operations with the help of the SRx-Server as intended.

Initially, we wanted to use the SRx-Proxy, a C implementation of the SRx API, implemented by NIST, that handles the communication between a router implementation and the SRx-Server. Unfortunately, integrating the C-based API into a Go-based project was not possible. With the support of NIST staff, we developed our own GoSRx proxy implementation that supports hello messages for session establishment, validation requests, and synchronization requests.

V. Live Demonstration

During our demonstration, we will show how our GoBGPSRx implementation processes various scripted BGP announcements and performs decisions based on path plausibility algorithms. Using the topology displayed in Figure 3, we will present a testbed in which we will send different BGP messages to the GoBGPSRx router to trigger reactions for multiple use cases. In detail, we will present the handling of three attacks: BGP prefix hijack, path shortening, and route leak. We will show that our implementation of path plausibility algorithms cannot detect BGP prefix hijacks but can detect and mitigate path-shortening attacks as well as route leaks.

VI. Conclusion

In this paper, we explained our implementation of two path plausibility algorithms, namely ASPA and AS-Cones, into the GoBGP routing daemon. By extending the NIST BGP-SRx software suite, we validated our implementation in a testbed capable of generating artificially crafted BGP announcements. During the evaluation, we showed that our implementation of ASPA and AS-Cones could detect path-shortening attacks and route leaks. Our contributions support the inter-domain routing community to further evaluate whether path plausibility algorithms can be a meaningful addition to the security of the inter-domain routing infrastructure.

Acknowledgment

We thank the anonymous reviewers for their feedback and Kai Hamich for his work on AS-Cones. Moreover, we thank Kyehwan Lee and the members of the NIST Internet Technologies Research Group [15] for their time for discussion. This work was partly supported by the German Federal Ministry of Education and Research (BMBF) within the project *6G-life*.

References

[1] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, “A Survey of BGP Security Issues and Solutions,” *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2009.
[2] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill, “BGP Hijacking Classification,” in *2019 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2019, pp. 25–32.
[3] M. S. Siddiqui, D. Montero, M. Yannuzzi, R. Serral-Gracia, and X. Masip-Bruin, “Route Leak Identification: A Step Toward Making Inter-Domain Routing More Reliable,” in *10th DRCN Conference*. IEEE, 2014, pp. 1–8.
[4] S. L. Murphy, “BGP Security Vulnerabilities Analysis,” RFC 4272, Jan. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4272
[5] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein, “BGP Prefix Origin Validation,” IETF, RFC 6811, January 2013.
[6] M. Lepinski and K. Sriram, “BGPsec Protocol Specification,” RFC 8205, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8205.txt
[7] A. Azimov, E. Bogomazov, R. Bush, K. Patel, J. Snijders, and K. Sriram, “BGP AS_PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects,” Internet Engineering Task Force, Internet-Draft draft-ietf-sidrops-aspa-verification-16, Aug. 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/16/
[8] J. Snijders, M. Stucchi, and M. Aelmans, “RPKI Autonomous Systems Cones: A Profile To Define Sets of Autonomous Systems Numbers To Facilitate BGP Filtering,” Internet Engineering Task Force, Internet-Draft draft-ietf-grow-rpki-as-cones-02, Apr. 2020. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-grow-rpki-as-cones-02
[9] T. Tauber, J. Snijders, and T. Bruijnzeels, “NANOG Discussion on ASPA Deployment,” https://seclists.org/nanog/2023/May/220.
[10] N. Hoeger, N. Rodday, and K. Hamich, “GoBGPSrx Repository,” https://github.com/nrodday/CNSM-23-demo.
[11] O. Borchert, K. Lee, K. Sriram, D. Montgomery, P. Gleichmann, and M. Adalier, “BGP Secure Routing Extension (BGP-SRx): Reference Implementation and Test Tools for Emerging BGP Security Standards,” National Institute of Standards and Technology, Tech. Rep. 2060, 2021. [Online]. Available: https://csrc.nist.gov/pubs/tn/2060/final
[12] R. Bush and R. Austein, “The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 2,” Internet Engineering Task Force, Internet-Draft draft-ietf-sidrops-8210bis-10, Jun. 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sidrops-8210bis/10/
[13] “Secure Router Extension (SRx) Proxy Protocol,” https://github.com/usnistgov/NIST-BGP-SRx/tree/master/srx-server/doc.
[14] K. Lee, “GoBGPsec,” https://github.com/usnistgov/gobgpsrx.
[15] “NIST Internet Technologies Research Group,” https://www.nist.gov/ctl/wireless-networks-division/internet-technologies-research-group.