System Detecting Border Gateway Protocol Anomalies Using Local and Remote Data

A. Starczewska, A. Nawrat, K. Daniec, J. Homa, K. Hołda

Abstract**—**Border Gateway Protocol (BGP) is the main routing protocol that enables routing establishment between all autonomous systems, which are the basic administrative units of the internet. Due to the poor protection of BGP, it is important to use additional BGP security systems. Many solutions to this problem have been proposed over the years, but none of them have been implemented on a global scale. This article describes a system capable of building images of real-time BGP network topology in order to detect BGP anomalies. Our proposal performs a detailed analysis of BGP messages that come into local network cards supplemented by information collected by remote collectors in different localizations.

Keywords**—**Border Gateway Protocol, BGP, BGP hijacking, cybersecurity, detection.

I. Introduction

INTERNET infrastructure consists of units called Autonomous Systems (AS) that exchange routing information with their peers using BGP. The current 4th version of BGP was first described in RFC 4271 [1] in 2006, and since then, the main structure of protocol has not changed. Unfortunately, it is a protocol vulnerable by design, which means that authors did not design any security algorithms. This fact is the reason of many BGP attacks causing breaks in the availability of the attacked services or interception of network traffic including sensitive data. Another issue is that the entire BGP configuration is written manually, which may be the source of many human errors and misconfigurations that could have the same effects as an intended attack.

Designing a universal BGP attack, the detection system is still an open issue; nevertheless, there are many publicly available web applications and tools that collect, analyze and provide current BGP data via UI or API. One of them is RIPE RIS (Routing Information Service belonging to the RIPE Network Coordination Center) [2], which consists of elements such as:

- RIS MRT files [3] – a dataset of all BGP messages collected by more than 20 RIS Route Collectors (RCC) for over 20 years in MRT format, which need one of the parsers listed at [3] to be correctly read.
- RIS Live [4] – a service providing BGP messages from RCC-s in real time via JavaScript and Python API-s or via streaming interface,
- RISwhois [5] – an application retrieving data about current prefix announcers,
- RIPEstat [6] – a service collecting current and historical BGP information, such as geolocalization, AS paths, AS neighbors, and RPKI status etc., available via UI or API basing on HTTP requests.

There are also other institutions apart from RIPE NCC that deal with BGP analysis. The University of Oregon developed a project called RouteViews [7] that collects BGP data from more than 30 collectors located around the world. Another is BGPMon from Cisco [8], which is an application monitoring users' prefixes in hundreds of locations and alerting in the case of odd path change. Besides, it performs RPKI validation. The Center for Applied Internet Data Analysis (CAIDA) develops an open-source framework called BGPStream [9], [10] that supports C++ and Python implementations of BGP analysis systems using RIPE RIS [2], RouteViews [7] and local archives.

The subject of the article is:

- an overview of existing systems and applications detecting BGP attacks,
- a description of the new application that analyzes BGP traffic in order to issue an anomaly alert.

II. State of the Art

A. Literature Analysis

Sermpezis et al. [11] described a real-time system called ARTEMIS that monitors the visibility of owned prefixes from the point of view of other AS-es using RIPE RIS [2], RouteViews [7], BGPMon [8] and BGPStream [10]. The system detects hijacking by origin modification and hijacking by path manipulation, both exact prefixes and subprefixes. The detection process is performed based on the list of monitored prefixes and AS-es owning them, and the list of monitored AS-es and their neighbors. After an attack occurrence, ARTEMIS runs one of its mitigation methods. According to [11], the system is able to mitigate the hijacking in just one minute.

Shi et al. [12] proposed a BGP hijacking detection system called Argus. It compares data retrieved from BGPMon [8] and RouteViews [7] mentioned in Section I with IP-s collected from Caida Ark [13], iPlane [14] and DNS records [15]. To confirm a hijacking, the system uses a number of so-called "Eyes of Argus", which are public route-servers and looking-glasses. Argus checks the suspicious prefix on the eyes' BGP routing tables and performs a ping test to validate the reachability of the prefix from each eye.

A. Starczewska, A. Nawrat, K. Daniec, J. Homa, and K. Hołda are with the Silesian University of Technology Faculty of Automatic Control, Electronics and Computer Science, 41100 Gliwice, Akademicka 16, Poland (e-mail: Alicja.Starczewska@polsl.pl, Aleksander.Nawrat@polsl.pl, Krzysztof.Daniec@polsl.pl, Jaroslaw.Homa@polsl.pl, Kacper.Hołda@polsl.pl).

Lad et al. [16] described PHAS – Prefix Hijack Alert System. It is a web service, where users define their prefixes on their accounts. The system monitors prefixes declared by users using data from RouteViews [7] and RIPE [4], and in the case of an anomaly, it sends an email to the user. In order to limit the number of alerts, it uses a time-window. To enable users to automate receiving alerts, authors provide local application for processing notification emails that can be customized.

Qiu et al. [17] proposed a system to detect bogus routes. Its work is divided into the initialization phase with given time, when it collects data to build the knowledge base, and the main phase, when it simultaneously receives routes and checks if they are bogus. The source of received routes depend on the deploy scenario. If the system is deployed by a service provider, it collects messages from its routers; however, it can also be used as monitoring system for whole internet and then it analyzes data from RouteViews [7] or RIPE RIS [2]. It stores every pair of upstream and downstream AS-es from updates' paths received during time window and every pair of prefix and origin AS. In the main phase, it checks if a new path consists of stored pairs and if the new pair prefix-AS is present in the stored pairs. If not, the route is classified as bogus.

Zheng et al. [18] described a scheme for detecting BGP hijacking basing on a number of monitors located around the world. The system periodically measures the maximum value of distances between each monitor and monitored prefix. If this value is big, prefix may be hijacked and path disagreement detection is performed. This process consists in comparing path to prefix with path to reference point, that should be chosen as close to the prefix as possible. The path to the reference point should be a part of or very similar to path to the prefix. If it is not, prefix is classified as hijacked.

B. Open Source Solutions

There are many open-source applications detecting BGP anomalies. One of them is BGPalerter [19], an application implemented in JavaScript language. It monitors chosen prefixes and AS-es for event such as lost visibility or hijack prefixes, invalid RPKI state or unexpected neighbors. The system uses data from RIS Live [4], RIPEstat [6] and a tool called rpki-validator [20] by the same author, that checks the RPKI state in services rpki-client.org [21], Routinator [22] and cfrpki from Cloudflare [23].

Another one is called BGPAA [24] and it works basing on archival MRT files [3]. That archive is updated every 5 minutes, which allows for analyzing both historical attacks and current data. The application detects BGP hijackings and presents results not only in log files, but also as graphs containing all routes with highlighted attacks. It is implemented using a framework called TaBi [25] created to facilitate BGP anomalies detection and classification. It needs the MRT files parser, for example MaBo [26] or BGPReader [27] that is a part of CAIDA's BGPStreamer mentioned in Section I.

Repository route_leaks [28] from ANSSI (fr. Agence nationale de la sécurité des systèmes d'information) contains different implementations detecting route leaks anomalies. The first algorithm compares the number of announced prefixes during one day and the number of conflicts between AS-es during this day. It is written in Python language and in order to speed up operation in Rust language. Another method implements SVM classifier. The last one is based on the proposal of Ju et al. [29] that filters BGP messages rejecting a few cases, e.g., if AS has announced a prefix form more than one day in the past year, it is IXP prefix or who is service says that AS-es announcing prefix belong to the same organization. For the rest of updates, if the number of AS-es announcing the same prefix exceeds given threshold it is classified as route leak. Data used by implementations are generated using TaBi [25] and MaBo [26] mentioned earlier.

III. Proposed System

The described solution is a multidimensional system designed to efficiently process incoming BGP data, meticulously analyze BGP updates and trigger real-time alerts in case of irregularities or suspicious activities. Its scheme is shown in Fig. 1. The main element of the system is the Node.js application that leverages event-driven design and non-blocking I/O operations. It consists of two modules. The first of them, analysis module, processes each received BGP message saved in JSON format in order to extract critical information. The preprocessing phase involves extracting data such as the message type, AS path, involved prefixes and timestamp. The basis of the algorithm is custom-defined class called ASN encapsulating crucial information about given AS, such as its number, associated prefixes and routing policies. Thanks to the array of objects of ASN type the system gains a comprehensive understanding of the BGP network's topology and routing relationships, that enables to intelligently analyze BGP updates within the broader context of AS interactions.

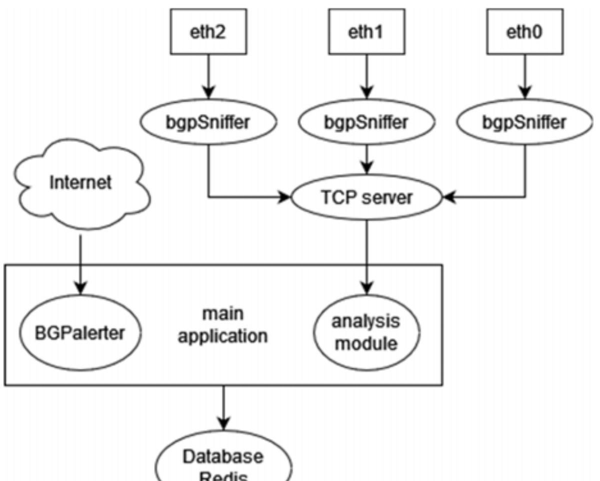

Fig. 1 System diagram

The other module is implemented basing on BGPalerter [19] mentioned in Section II *B*. It is responsible for monitoring global BGP infrastructure visible from different collectors located all around the world. It performs that functionality using publicly available services, such as RIS Live [4], RIPEstat [6], NTT RPKI VRPs [22] containing information related to historical AS entry records, Cloudflare VRPs [23] providing insights into the number of ROAs over time, along with statistics on trust anchors and maximum prefix lengths and Rpki-client.org VRPs [21] querying the global RPKI repository system, verifying untrusted network input. BGPalerter implements 10 different monitor analyzing data receiving from above services but it is focused on chosen AS. At the beginning of work application asks user about AS-es, that should be monitored. Then the application downloads information about given AS from public services. This information mainly contains a list of announced prefixes. Unfortunately, in original BGPalerter that list is not always valid. Services used by application are based on data from collectors. If a request is sent during an attack or other anomaly, collectors may see an improper prefix list and, as a result of that situation, the received prefix list may be false. However, this issue will be fixed in the future work.

Both modules, analysis module and BGPalerter, generate a set of alerts regarding e.g., suspicious prefixes or invalid RPKI. A full list of alerts is presented in Table I. Generated alerts are parsed by an extremely fast msgpackr library [30] into a compact binary format. Next, they are inserted into the Redis [31] database.

TABLE I
ALERTS REPORTED BY SYSTEM

Alert	Description
MANY NOTLISTED PREFIXES	AS announces prefixes it should not announce.
ONE NOTLISTED PREFIX	AS announces one prefix it should not announce.
CHANGED AS PREFIX	Prefix is announced by other AS then previous.
NEW PREFIX	AS announces a new prefix.
PREFIX LENGTH	Wrong prefix length.
NEW NEIGHBOUR	There is a new neighbor.
ROA DISAPPEAR	ROA disappeared from TA.
ROA EXPIRING	ROA is going to expire in given time.
ROA CHANGE	ROA is changed.
NO LONGER ROA ROUTE	Path is not in ROA, but it was.
NOT ROA ROUTE	Path is not in ROA.
NOT RPKI VAL ROUTE	Path is not RPKI valid.
PREFIX VISIBILITY	Prefix is not visible by peer.
NEW PREFIX	New prefix is announced.
PATH CHANGED	Path to the AS is changed.
OLD PREFIX ANNOUNCED	Withdrawn prefix is announced again.
LD PREFIX DIFFERENT AS	Withdrawn prefix is announced by another AS.
NO CONNECTION	There is no connection with peer.

Another element of the system is the TCP server being the entry point of the application, serving as the central hub that accepts incoming BGP update messages from multiple connected clients. This module efficiently manages client connections, establishes communication channels, and routes the incoming data to the appropriate processing components. Leveraging Node.js's event-driven architecture, the server module is adept at handling concurrent connections without compromising performance, thereby ensuring seamless data flow and timely processing.

The next part of the system is the program called bgpSniffer that acts as the client for TCP server. The system allows to collect BGP messages from multiple network cards belonging to the system's subnet. For each network card there is one running instance of bgpSniffer. The program captures incoming packets from the selected network card and filters them for BGP messages. Then, it converts the payload into JSON format and transmits returned data into the TCP server. It is implemented in C++ language and uses the libpcap library.

IV. TEST STAND

The system was tested using real data. A scheme of the test stand is presented in Fig. 2. It consists of four Ubuntu virtual machines with running Bird [32], one of the BGP implementations for Linux. These machines are connected via virtual switches and are configured as peers with ASN from a private range. One of them acts as the private peer of global AS8508 of the Silesian University of Technology and receives the whole BGP traffic. Then, it transmits the routing table to other private peers. The described application is running at AS65530 and captures data from three network cards connected to AS65531, AS65532 and AS6533.

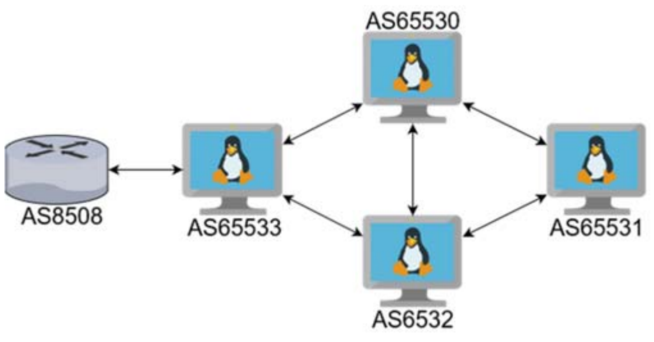

Fig. 2 Test stand

The application was tested in terms of the number of generated alerts. The test scenario assumed executing the program for 5 hours and saving the number of generated alerts and their size in the Redis database every hour. Each module was tested separately. First, there was a test of the analysis module with three instances of bgpSniffer for three network cards and any AS configured as input of the BGPalerter. After the 5 hours, the application was executed the second time without any instances of bgpSniffer and with AS8508 configured as the input of BGPalerter. The results are shown in Table II. It was noticed that the analysis module generates many more alerts than the BGPalerter. The reason for this situation is that the analysis module monitors every single incoming prefix and BGPalerter focuses on user prefixes. Most of the information generated by the analysis module is used to build the BGP network topology, which is highly dynamic and paths to different origins change very often. This phenomenon is visible in the large number of alerts. On the other side, BGPalerter does not check paths to the origin, so that part of the application generates far fewer alerts than the analysis module. What is important is that despite the large number of alerts saved in the database, their size is relatively small due to the good data compression achieved with msgpackr. This is a significant advantage of the system due to limited available storage in these types of applications.

The other important fact is that the analysis module generates only about four alerts per second. Considering that the system receives dozens of BGP messages per second, that amount is relatively small, which indicates that the system may be computationally inefficient.

TABLE II
TEST RESULTS: NUMBER OF ALERTS AND THEIR SIZE

Time	Analysis module	BGPalerter
1 h	10892 / 5.18 MB	4 / 862.70 kB
2 h	25368 / 10.75 MB	13 / 865.90 kB
3 h	37281 / 17.62 MB	21 / 871.13 kB
4 h	51124 / 24.29 MB	34 / 886.55 kB
5 h	68237 / 32.38 MB	42 / 900.81 kB

V. CONCLUSION

This article describes a proposed system analyzing a BGP network and detecting suspicious BGP data. The application monitors visibility of owned prefixes using publicly available services with data from collectors in different localizations, checks agreement of incoming data with RPKI and builds current BGP topology seen by itself. Unfortunately, tests showed the need to increase capacity, which is the goal of future development of the application. Despite that, the system is still a useful tool for BGP analysis that is able to build current BGP network topology and detect BGP anomalies.

ACKNOWLEDGMENT

This work has been supported by European Union - European Regional Development Fund as a project ID: POIR.01.01.01-00-0029/20 “Development of an innovative platform based on blockchain technology”.

REFERENCES

[1] REKHTER, Yakov; LI, Tony; HARES, Susan (ed.). RFC 4271: A border gateway protocol 4 (BGP-4). 2006.
[2] „RIPE NCC Routing Information Service” (Online). Available on: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris
[3] „RIS Docs: Route Collection Raw Data: MRT Files” (Online). Available on: https://ris.ripe.net/docs/20_raw_data_mrt.html
[4] „Routing Information Service Live” (Online). Available on: https://ris-live.ripe.net/
[5] „RIS Docs: RISwhois” (Online). Available on: https://ris.ripe.net/docs/27_riswhois.html#dataservice
[6] „RIPEstat: Providing open data and insights for Internet resources” (Online). Available on: https://stat.ripe.net/about/
[7] „University of Oregon RouteViews Project” (Online). Available on: https://www.routeviews.org/routeviews/
[8] „BGPMon is Now Part of CrossworkCloud” (Online). Available on: https://www.bgpmon.net/
[9] ORSINI, Chiara, et al. BGPStream: a software framework for live and historical BGP data analysis. In: *Proceedings of the 2016 Internet Measurement Conference.* 2016. p. 429-444.
[10] „BGPStream” (Online). Available on: https://bgpstream.caida.org/
[11] SERMPEZIS, Pavlos, et al. ARTEMIS: Neutralizing BGP hijacking within a minute. *IEEE/ACM Transactions on Networking*, 2018, 26.6: 2471-2486.
[12] SHI, Xingang, et al. Detecting prefix hijackings in the internet with argus. In: *Proceedings of the 2012 Internet Measurement Conference.* 2012. p. 15-28.
[13] „Archipelago (Ark) Measurement Infrastructure” (Online). Available on: https://www.caida.org/projects/ark/
[14] MADHYASTHA, Harsha V., et al. iPlane: An information plane for distributed services. In: *Proceedings of the 7th symposium on Operating systems design and implementation*. 2006. p. 367-380.
[15] „Hurricane Electric Internet Services” (Online). Available on: https://bgp.he.net/net/166.111.0.0/16#_dns
[16] LAD, Mohit, et al. PHAS: A Prefix Hijack Alert System. In: *USENIX Security symposium.* 2006. p. 3.
[17] QIU, Jian, et al. Detecting bogus BGP route information: Going beyond prefix hijacking. In: *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007*. IEEE, 2007. p. 381-390.
[18] ZHENG, Changxi, et al. A light-weight distributed scheme for detecting IP prefix hijacks in real-time. *ACM SIGCOMM Computer Communication Review*, 2007, 37.4: 277-288.
[19] „BGPalerter” (Online). Available on: https://github.com/nttgin/BGPalerter
[20] „rpki-validator” (Online). Available on: https://github.com/massimocandela/rpki-validator
[21] „rpki-client” (Online). Available on: https://www.rpki-client.org/
[22] „RPKI TOOLS: Routinator” (Online). Available on: https://www.nlnetlabs.nl/projects/rpki/routinator/
[23] „Cloudflare RPKI Validator Tools and Libraries” (Online). Available on: https://github.com/cloudflare/cfrpki
[24] „BGPAA” (Online). Available on: https://github.com/BGPAA/BGP_Attack_Analysis
[25] „TaBi – Track BGP Hijacks” (Online). Available on: https://github.com/ANSSI-FR/tabi
[26] „MaBo – MRT and BGP in OCaml” (Online). Available on: https://github.com/ANSSI-FR/mabo
[27] „BGPStream: BGPReader” (Online). Available on: https://bgpstream.caida.org/docs/tools/bgpreader
[28] „Route Leak Detection” (Online). Available on: https://github.com/ANSSI-FR/route_leaks
[29] JU, Qing; KHARE, Varun; ZHANG, Beichuan. Large route leak detection. *NANOG'49*, 2010.
[30] „Namby Pamby Magicians: msgpackr” (Online). Available on: https://www.npmjs.com/package/msgpackr
[31] „Redis” (Online). Available on: https://redis.io/
[32] „The BIRD Internet Routing Deamon” (Online). Available on: https://bird.network.cz/