BPHS: A BGP Prefix Hijacking Simulation Tool Supporting RPKI filtering

Georgios Eptaminitakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Prof. *Xenofontas Dimitropoulos*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

BPHS: A BGP Prefix Hijacking Simulation Tool Supporting RPKI filtering

Thesis submitted by
Georgios Eptaminitakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: [illegible]
Georgios Eptaminitakis

Committee approvals: [illegible]
Xenofontas Dimitropoulos
Associate Professor, Thesis Supervisor

[illegible]
Evangelos Markatos
Professor, Committee Member

[illegible]
Kostas Magoutis
Associate Professor, Committee Member

Departmental approval: [illegible]
Polyvios Pratikakis
Associate Professor, Director of Graduate Studies

Heraklion, March 2022

BPHS: A BGP Prefix Hijacking Simulation Tool Supporting RPKI filtering

Abstract

The *Border Gateway Protocol* (BGP) is used by the Autonomous Systems (e.g., Comcast, AT&T, COSMOTE) to advertise AS paths for the corresponding routed IP address space (i.e., IPv4/IPv6 network prefixes) and establish inter/intra-domain routes in the Internet. Despite its scalability and capabilities of expressing complex routing policies, BGP lacks many security features by design, like the authentication of the advertised routes. Thus, an Autonomous System (AS) is able to advertise illegitimate routes for IP prefixes that it does not own. These advertisements propagate and "pollute" many ASes, or even the entire Internet, affecting service availability, integrity, and confidentiality of communications. This phenomenon is called *BGP prefix hijacking* and can be caused by either router misconfigurations or malicious attacks.

The *Resource Public Key Infrastructure* (RPKI) is a hierarchical certification system that aims to protect the Internet against these BGP prefix hijacking attacks, introducing IP prefix ownership authentication. Despite its crucial role in Internet security, RPKI deployment is slow (i.e., around 20% of the total ASes on the Internet) due to its limited adoption from most ASes (i.e., it protects an AS only if a large number of other ASes already use it) and due to its complex mechanisms resulting in human errors.

Several research works have tried to measure the impact of the BGP prefix hijacking attacks and the benefits of the RPKI adoption in the Internet through BGP simulation algorithms that model the Internet graph and the BGP protocol. However, there is not any related work proposing a BGP simulator enabling network operators to quickly and easily assess the vulnerability of their Autonomous Systems to BGP prefix hijacking attacks and conduct their study through a user-friendly and plug&play BGP simulation tool or service.

In this thesis, we introduce *BPHS*, the first BGP Prefix Hijacking Simulation tool that enables network operators to quickly and easily (a) assess the vulnerability of their Autonomous Systems to BGP prefix hijacks and (b) measure the benefits of the RPKI's adoption in the Internet, through a user-friendly web application.

We evaluate BPHS by replaying real historical hijacks detected in the Internet and reveal the benefits of the RPKI adoption by conducting a comparative study (using BPHS) showing the attacker's success rate for different RPKI filtering scenarios. In addition, we extract two rankings lists showing the most vulnerable ASes and countries to BGP prefix hijacking attacks, including the most vulnerable Greek ASes. The evaluation results show that BPHS estimates the impact of a hijack with high accuracy and that the RPKI filtering in the Internet backbone significantly reduces the BGP attacks.

BPHS: Ένα εργαλείο προσομοίωσης επιθέσεων προθέματος BGP που υποστηρίζει φιλτράρισμα RPKI

Περίληψη

Το πρωτόκολλο *BGP (Border Gateway Protocol)* χρησιμοποιείτε από τα Αυτόνομα Συστήματα (όπως, Comcast, AT&T, COSMOTE) ώστε να διαφημίζουν στο Internet μονοπάτια δρομολόγησης για το σύνολο του χώρου IP διευθύνσεων (δηλαδή, IPv4/IPv6 προθέματα δικτύου) και να εγκαθιδρύουν inter/intra-domain διαδρομές στο Internet. Παρά την επεκτασιμότητα και την δυνατότητα να εκφράζει πολύπλοκες πολιτικές δρομολόγησης, το BGP (απο το σχεδιασμό του) δεν διαθέτει κανένα μηχανισμό ασφάλειας, όπως είναι η πιστοποίηση των διαφημιζόμενων διαδρομών. Έτσι, ένα Αυτόνομο Σύστημα έχει την δυνατότητα να διαφημίζει παράνομες διαδρομές για προθέματα IP που δεν κατέχει. Αυτές οι παράνομες ανακοινώσεις διαδίδονται και μολύνουν πολλά Αυτόνομα Συστήματα ή και ακόμα ολόκληρο το Internet, με αποτέλεσμα να επηρεάζουν τη διαθεσιμότητα, την ακεραιότητα και το απόρρητο των επικοινωνιών. Το φαινόμενο αυτό ονομάζεται *πειρατεία προθέματος BGP (ή BGP prefix hijacking)* και μπορεί να προκληθεί είτε απο λάθος παραμετροποιήσεις δρομολογητών, είτε από κακόβουλες επιθέσεις.

Το *RPKI (Resource Public Key Infrastructure)* είναι ένα ιεραρχικό σύστημα πιστοποίησης που στοχεύει στην προστασία του Internet από αυτές τις επιθέσεις πειρατείας προθέματος BGP, εισάγοντας την πιστοποίηση ιδιοκτησίας προθέματος IP. Παρά τον σημαντικό του ρόλο στην ασφάλεια του Internet, η ανάπτυξη του RPKI είναι αργή (χρησιμοποιείτε περίπου απο το 20% των Αυτόνομων Συστημάτων στο Internet) λόγω της περιορισμένης υϊοθέτησης του απο τα περισσότερα Αυτόνομα Συστήματα (δηλαδη, προστατεύει ένα Αυτόνομο Σύστημα μόνο όταν ένας μεγάλος αριθμός απο Αυτόνομα Σύστηματα το χρησιμοποιεί ήδη) και λόγω των πολύπλοκων μηχανισμών του που οδηγούν σε ανθρώπινα λάθη.

Αρκετές ερευνητικές εργασίες έχουν προσπαθήσει να εκτιμήσουν τον αντίκτυπο των επιθέσεων πειρατείας προθέματος BGP και τα οφέλη της υιοθέτησης του RPKI στο Internet μέσω αλγορίθμων προσομοίωσης που μοντελοποιούν το γράφημα του Internet και το πρωτόκολλο BGP. Ωστόσο, δεν υπάρχει καμία σχετική εργασία που να προτείνει έναν προσομοιωτή BGP ο οποίος να επιτρέπει στους διαχειριστές δικτύων να αξιολογούν γρήγορα και εύκολα την ευπάθεια των Αυτόνομων Συστημάτων τους σε επιθέσεις πειρατείας προθέματος BGP και να διεξάγουν την έρευνα τους μέσω ενός φιλικού προς τον χρήστη εργαλείου (ή υπηρεσίας) προσομοίωσης BGP.

Σε αυτή την εργασία, εισάγουμε το *BPHS*, το πρώτο εργαλείο προσομοίωσης επιθέσεων πειρατείας προθέματος BGP που επιτρέπει στους διαχειριστές δικτύων (α) να αξιολογούν την ευπάθεια των Αυτόνομων Συστημάτων τους σε επιθέσεις προθέματος BGP και (β) να μετρούν τα οφέλη της υιοθέτησης του RPKI στο Internet, γρήγορα και εύκολα, μέσω μιας φιλικής προς τον χρήστη web εφαρμογής.

Αξιολογούμε το BPHS αναπαράγοντας πραγματικές ιστορικές επιθέσεις προθέματος BGP που εντοπίστηκαν στο Διαδίκτυο και αποκαλύπτουμε τα οφέλη της υιοθέτησης του RPKI πραγματοποιώντας μια συγκριτική μελέτη (χρησιμοποιώντας το BPHS) που δείχνει το ποσοστό επιτυχίας των επιθέσεων για διαφορετικά σενάρια φιλτραρίσματος RPKI. Επιπλέον, εξάγουμε δύο λίστες κατάταξης που δείχνουν τα πιο ευάλωτα Αυτόνομα Συστήματα και τις πιο ευάλωτες χώρες σε επιθέσεις πειρατείας προθέματος BGP, συμπεριλαμβανομένων των πιο ευάλωτων Ελληνικών Αυτόνομων Συστήματων. Τα αποτελέσματα της αξιολόγησης δείχνουν ότι το BPHS εκτιμά τον αντίκτυπο των επιθέσεων με υψηλή ακρίβεια και ότι το φιλτράρισμα RPKI στην ραχοκοκαλία του Internet μειώνει σημαντικά τις επιθέσεις προθέματος BGP.

Ευχαριστίες

Μέσα απο την εκπόνηση της συγκεκριμένης εργασίας συνειδητοποίησα πόσο σημαντικό είναι να έχεις δίπλα σου ανθρώπους που μπορούν να σε στηρίζουν σε κάθε σου βήμα. Οπότε θα ήθελα να ευχαριστήσω απο καρδιάς:

- Τον επόπτη καθηγητή μου, κύριο Ξενοφώντα Δημητρόπουλο, για την εμπιστοσύνη που μου έδειξε καθ΄ όλη την διάρκεια της εργασίας αλλά και για την δυνατότητα που μου έδωσε να γίνω μέλος του INSPIRE Group.

- Τον Γιώργο Νομικό, ερευνητής στο INSPIRE Group ειδικός σε θέματα μετρήσεων διαδικτύου, ο οποίος με βοήθησε καθοριστικά σε θέματα οργάνωσης και τεχνικής συγγραφής της εργασίας.

- Τον Αλέξανδρο Μηλολιδάκη, διδακτορικός φοιτητής στο KTH, ο οποίος με εισήγαγε ερευνητικά στο θέμα της εργασιας και μου έδωσε το έναυσμα να κατανοήσω τις σχετικές προκλήσεις της.

- Όλα τα μέλη, και ειδικότερα για εμένα φίλους - Αντώνη Χατζηβασιλείου, Βασίλη Πετρόπουλο, Μανώλη Λακιωτάκη, Αντρέα Σίσκο, Κώστα Αρακαδάκη - , του INSPIRE Group, για την άψογη συνεργασία μας και το υγειές εργασιακό κλίμα.

- Την οικογένειά μου, η οποία με εμψυχώνει πάντα στις δύσκολες στιγμές και με υποστηρίζει σε ότι και αν κάνω σ΄ αυτή τη ζωή.

Τέλος, θα ήθελα να ευχαριστήσω τον κύριο Βαγγέλη Μαρκάτο και τον κύριο Κώστα Μαγκούτη, καθηγητές στο Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης, οι οποίοι δέχθηκαν με μεγάλη χαρά να είναι μέλη της επιτροπής εξέτασης της εργασίας μου.

Στην οικογένειά μου

Table of contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The Internet, is a network of networks, which are connected with each other at various locations around the world for exchanging traffic and delivering services to other networks or end-users. Each one of these networks constitutes an Autonomous System (AS). The BGP protocol [115], enables ASes to send and receive reachability information (i.e., BGP routes) related to their IP prefixes (i.e., groups of IP addresses) and the paths leading to them.

Despite its scalability and capabilities of expressing complex routing policies, BGP lacking authentication of routes. As a result, an AS is able to advertise illegitimate routes for IP prefixes it does not own. These illegitimate advertisements propagate and "pollute" many ASes, or even the entire Internet, affecting service availability, integrity, and confidentiality of communications. This phenomenon, called BGP prefix hijacking, can be caused by router misconfiguration [67], [16] or malicious attacks [28], [114], [125].

To protect against prefix hijacks, the Internet Engineering Task Force (IETF) [63] promotes/suggests the deployment of the Resource Public Key Infrastructure (RPKI), which binds IP address blocks to "owner" ASes via cryptographic signatures [103]. RPKI enables ASes to validate that an AS advertising IP addresses in BGP is authorized to do so and thus to detect and discard prefix hijacks.

Despite its crucial role in Internet security, there are obstacles to RPKI's ubiquitous adoption: (a) certifying ownership of IP address blocks in RPKI is a manual and hierarchical process that requires coordination between ASes (i.e., an AS issues RPKI certificates for its customer ASes), (b) human error is common (e.g., invalid RPKI certificates for the 10% of the prefixes advertised in BGP), resulting a very slow RPKI adoption (i.e., around 20% of the total ASes on the Internet) [71].

The increase of BGP attacks in today's Internet, the unawareness of network operators about the vulnerability of their networks to BGP attacks, and the slow RPKI adoption have led the research community to develop simulation algorithms, that try to evaluate and mitigate these problems. Simulations are preferred over real attacks on the Internet, because we properly design and orchestrate a number of artificial BGP attacks in a secure environment extracting useful research insights that are impossible to be inferred through real experiments on the Internet due to infrastructure limitations and reachability concerns from the network operators (i.e., AS disconnection from the Internet). There are plenty of research works that tried to answer research questions about BGP and RPKI through BGP simulations [72], [86], [82], [118], but none of them have proposed a BGP simulator that enables network operators or naive users without programming skills to conduct their study through a user-friendly and plug&play BGP simulation tool or service.

In this work, **we introduce *BPHS***, *the first BGP Prefix Hijacking Simulation tool that enables network operators to quickly and easily (a) assess the vulnerability of their Autonomous Systems to BGP prefix hijacks and (b) measure the benefits of the RPKI's adoption on the Internet, through a user-friendly web application. With BPHS, the network operators can simulate different types of BGP hijacking attacks and obtain the simulation results through an automated and well-designed Graphical User Interface.*

1.2 Contributions

The first contribution of this thesis is a BGP Prefix Hijacking Simulation Tool, called BPHS. BPHS is the first BGP simulator that developed as a full stack web application. It supports (a) user-friendly GUI for easy interaction from desktop and mobile web-browsers, (b) multi-threading execution enabling end-users to retrieve quicker results per simulation, (c) REST API [107] to allow other applications to communicate with the simulator and (d) realtime RPKI filtering using the most up-to-date data from the RPKI databases, for more realistic simulation results. BPHS, models the Internet graph through the well-known AS relationship datasets from CAIDA [14], applies the user preferences on the generated graph (e.g., random or custom simulation, hijack type, RPKI filtering mode) and finally simulates the BGP protocol along with different types of prefix hijacks.

The second contribution of this work is an experimental study showing the ability of BPHS to detect the hijacked ASes as they identified in real historical datasets of BGP hijacking events that happened on the Internet. The evaluation results showed a 70% success rate for prefix hijacking events and 87% for sub-prefix hijacking events on average.

The third contribution of the thesis is a comparative study showing the RPKI adoption benefits on the Internet's backbone. In our evaluation experiments, we measured the hijacker's success rate (i.e., the impact of the attack) for different RPKI adoption rates and use cases (e.g., RPKI filtering only from the top 100 ASes [6] or according to RPKI adoption status on today's Internet). The evaluation results showed a significant reduction in the attacker's success rate (30% for the prefix hijacks and 40% for the subprefix hijacks) when only a few ASes on the Internet backbone (e.g., the top 100 ASes) perform RPKI filtering.

The fourth contribution of this work is the ability to compute an AS - Country vulnerability ranking for each BGP hijacking simulation initiated by the user. During our experimental evaluation, we conducted thousands of successful simulations. Thus, we decided to compute an overall AS - Country vulnerability ranking list from all the simulations. BPHS ranked an AS belonging to the most popular streaming provider as the 3rd most vulnerable AS on the Internet and two Greek ASes in the top 600 of the list. Finally, our algorithm ranked the United States as the most vulnerable country and surprisingly Greece ranked eleventh in the list.

1.3 Overview

First, in Section 2 (Background), we describe core topics related to the Internet routing and the BGP protocol; we further present in detail the BGP prefix hijacking problem and the main functionalities of the RPKI framework; also we present the Cooperative Association for Internet Data Analysis (CAIDA) which provides research datasets that helped us in simulator's development and evaluation.

In the 3rd section, we discuss the related research work on BGP hijacking simulation mechanisms, and in chapter 4, we analytically present the BPHS tool by describing the tool's architecture and the technologies that are used to support the BPHS's main components. Then, we present the main features of the simulator and finally we list the challenges of the BPHS's development.

In chapter 5, we present the evaluation studies that we conduct to (a) measure the ability of the BPHS to extract realistic simulation results, (b) show the RPKI adoption benefits on the Internet's backbone and (c) extract useful insights from a wide range of simulation events. Finally, in chapter 6 we conclude our thesis and we elaborate on directions for future work.

Chapter 2

Background

2.1 Internet Routing

The Internet is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) [78] to link devices worldwide. It is a network of networks in which users at any one computer which has an IP address can, if they have permission, to get information from any other computer which has an IP address. An IP address is a numerical label that is assigned to network devices that communicate in the Internet through the IP protocol, and is used to identify the device and address the location. An IP address might be PUBLIC where it is reachable from the whole Internet, or PRIVATE where it is only routable inside a private network.

An Autonomous System includes a collection of networking devices (e.g., routers, middleboxes, etc.) along with network resources (e.g., IP network prefixes) within the Internet. An IP prefix is an aggregation of IP addresses that is often written in Classless Inter-Domain Routing (CIDR) notation. In CIDR format we write first, the address of a network, followed by a slash character (/), and ending with the bit-length of the prefix (e.g., 130.10.0.0/21 for IPv4, ::ffff:820a:0/21 for IPv6). Network prefixes can be advertised by Autonomous Systems to the Internet or not.

To identify the Autonomous Systems, the Internet Assigned Numbers Authority (IANA) [30] allocates unique AS Numbers (ASN) to Regional Internet Registries (RIRs) [56] for each AS (e.g. Google [AS 15169], Comcast [AS 7922]). The unique AS number/identifier allows an AS to communicate with other ASes on the Internet. An AS can have a private ASN and a public ASN. A *private* ASN is a unique identifier that is used by ASes for private communication with a neighboring AS (e.g., an upstream Internet provider) via BGP. On the other hand, a *public* ASN is a unique identifier that is used by an AS to advertise its existence to the public Internet and is needed to exchange information over the Internet.

Autonomous Systems are interconnected and communicate with each other with two basic protocols: the Interior Gateway Protocol (IGP) and the Exterior Gateway Protocol (EGP). The Interior Gateway Protocol (e.g, OSPF [111], RIP [106], EIGRP [60]) is used for exchanging routing information within an autonomous system (i.e., intra-domain traffic), whereas the Exterior Gateway Protocol (BGP) is used to exchange routing information between autonomous systems (i.e., inter-domain traffic) and relies on IGPs to resolve routes within an autonomous system (see Figure 2.1).

2.2 Border Gateway Protocol (BGP)

The Border Gateway Protocol version 4 [91] (BGPv4) is a standardized External Gateway Protocol designed to exchange routing information between ASes in order to handle intra-domain and inter-domain traffic. It selects the most preferable path the data will go through. BGP uses the information in BGP routing tables to determine the next AS hop, which, viewed in the context of a chain of ASes between the source and destination of data, determines the route to its destination. By default, BGP will send traffic to the path with the shortest logical distance (fewest number of AS hops). This process is called BGP best path selection or AS path selection.

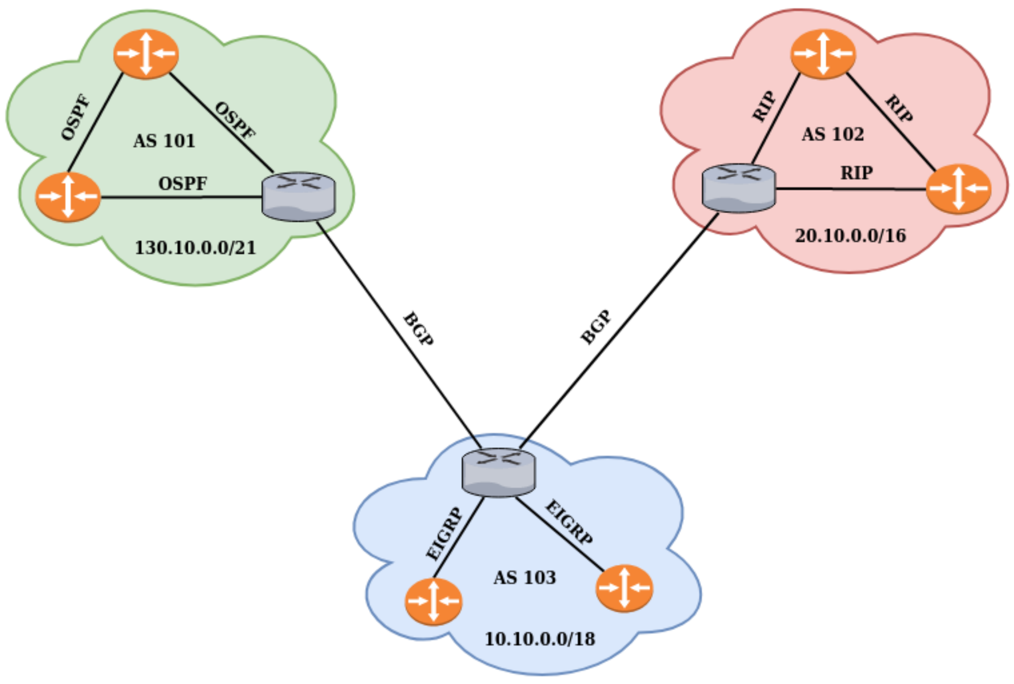


Figure 2.1: Internet Topology with Internal and External Gateway Protocols. AS101 announces prefix 130.10.0.0/21 to their BGP neighbors, AS102 announces prefix 20.10.0.0/16 to their BGP neighbors and Upstream AS103 announces prefix 10.10.0.0/18 to their BGP neighbors. Note that while all three ASes speak different IGPs internally, all of them use BGP for route exchanging.

2.2.1 BGP Selection Algorithm

Border Gateway Protocol routers typically receive multiple paths to the same destination. The BGP best path algorithm decides which is the best path to install in the IP routing table and to use for traffic forwarding. A list of best path selection criteria is listed below, sorted in order of preference [8]:

1. **Weight**
2. **Local Preference**
3. **Network or Aggregate**
4. **Shortest AS PATH**
5. **Lowest origin type**
6. **Lowest multi-exit discriminator (MED)**
7. **eBGP over iBGP**
8. **Lowest IGP metric**
9. **Multiple paths**
10. **External paths**
11. **Lowest router ID**
12. **Minimum cluster list**
13. **Lowest neighbor address**

From the above selection metrics, we focus on **Local Preference** and **shortest AS PATH**, because these two rules are used for the thesis needs.

2.2.2 Local Preference Attribute

Local Preference (aka *LOCAL_PREF*) is a BGP well-known attribute that is used when there are multiple exit paths out from a single AS. Whereas the *MED* is used by routers to determine the optimal route when traffic enters an AS, the *LOCAL_PREF* helps routers to determine the optimal route when traffic leaves an AS. The basic/core characteristics of the *LOCAL_PREF* BGP attribute are listed below:

1. Well-known and discretionary BGP attribute.
2. The path with the **highest local preference is preferred**.
3. Used for **outbound traffic filtering**.

4. Not exchanged between external BGP routers.
5. Sent to all internal BGP routers in the AS.
6. Default value is 100.

Suppose a router has three BGP sessions (e.g., see Figure 2.2), and receives a route towards a given destination over each session, where one route has a local preference of 200, one has a local preference of 150 and one has a local preference of 75. *"Prefer the path with the highest local preference"* means the route with a *LOCAL_PREF* of 200 is used, which means that it is installed in the main *routing table* and propagated to BGP neighbors whether allowed by the specified policies/filters. The other two routes with *LOCAL_PREFs* of 150 and 75 are kept in the *BGP table*, and also not propagated to BGP neighbors.

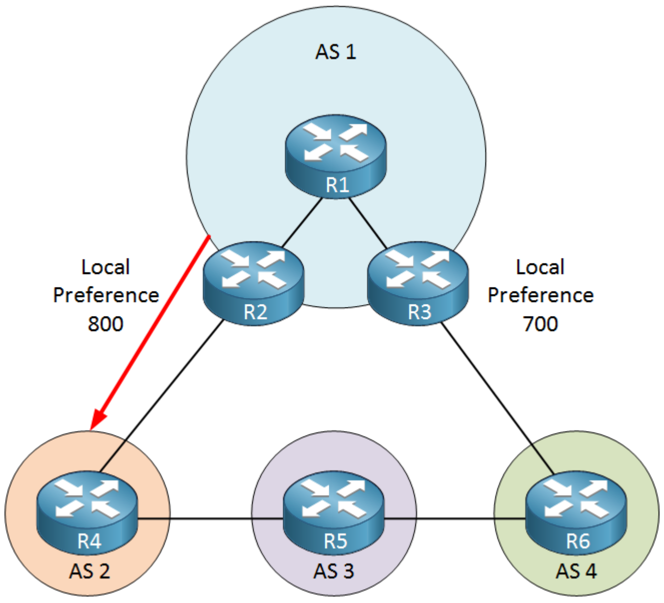


Figure 2.2: AS 1 send all outbound traffic towards AS 2 due to higher Local Preference on routes learned by router R2 [9].

.

2.2.3 Shortest AS PATH

When the BGP routing mechanism is not possible to select a route based on weight, local preference and the route does not originate from a local or aggregated network [36], then BGP decides for the best route based on the lowest AS hops towards the destination IP prefix. For example, if a BGP speaker receives multiple routes for an IP prefix, (a) AS1,AS2,AS3 and (b) AS1,AS4,AS2,AS3 and they do not have any other higher criterion to discriminate, then route (a) will always be preferred over the route (b) due to lowest AS hops.

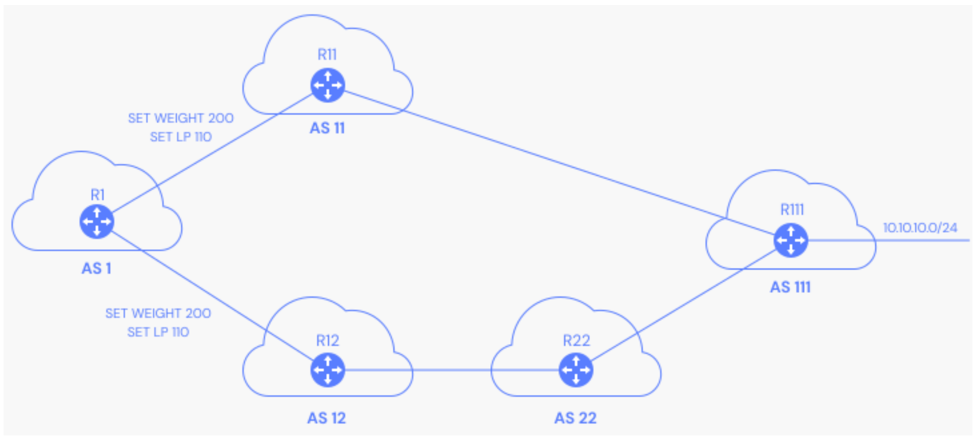
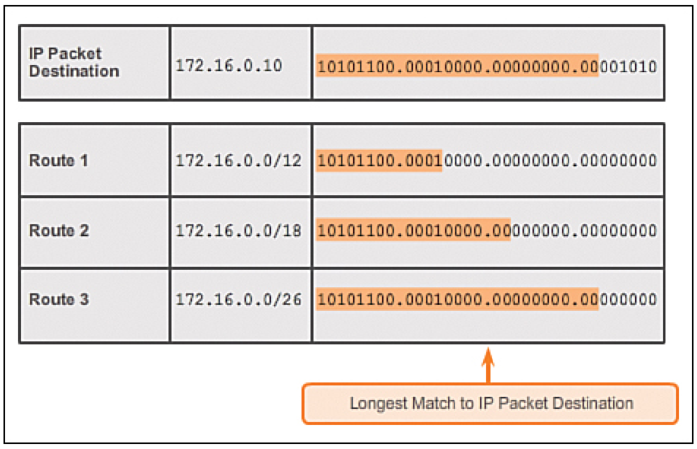


Figure 2.3: Shortest AS PATH example [7].

In Figure 2.3, the route received by router R1 from router R11 has the AS_PATH: [11 111] and the route received from router R12 has the AS_PATH: [12 22 111]. Because a shorter AS_PATH is preferred than a longer AS_PATH, if all the previous steps of the algorithm could not select the best route, the route with the shortest AS_PATH is selected as the best one.

2.2.4 Longest Prefix Match in routing

In order to determine the packet's next-hop interface, routers use the Longest Match Routing Rule (most specific routing table entry [17]), sometimes referred to as the longest prefix match or maximum prefix length match. The Longest Match Routing Rule is an algorithm used by routers to select an entry from a routing table. The router uses the longest (prefix) match to determine the egress (outbound) interface and the address of the next device to which to send a packet.



IP Packet Destination	172.16.0.10	10101100.00010000.00000000.00001010

Route 1	172.16.0.0/12	10101100.00010000.00000000.00000000
Route 2	172.16.0.0/18	10101100.00010000.00000000.00000000
Route 3	172.16.0.0/26	10101100.00010000.00000000.00000000

Longest Match to IP Packet Destination

Figure 2.4: Longest Prefix Match wins in routing decision.

When the router receives a packet, processes its header and compares the destination IP address, with the entries in the routing table. The entry that has the longest number of network bits that match the IP destination address is always the best match (or best path).

For example, in Figure 2.4, we see the routing table entries of a random router in a network topology. This router receives a packet with a destination IP address of 172.16.0.10 . Router (a) compares the destination IP address with the routing table entries, (b) identifies the longest match on prefix 172.16.0.0/26, and (c) forwards the packet to the interface towards "Route 3".

2.2.5 AS Relationships

Interdomain traffic engineering requirements are diverse and often motivated by the need to balance the traffic on links with other ASes and to reduce the cost of carrying traffic on these links. These requirements depend on the connectivity of an AS with others, but also on the type of business handled by this AS.

Connectivity between ASes is mainly composed of two types of relationships. The most frequent relationship between ASes is the ***customer-to-provider*** relationship, where a customer AS pays to use a link connected to its provider. This is this the most costly relationship of a customer AS (in terms of money per traffic). A *stub* AS (i.e., an AS without customers) usually tries to maintain at least two of these links for performance and redundancy reasons [124]. In addition, larger ASes typically try to obtain ***peer-to-peer*** relationships with other ASes and then share the cost of the link with the other AS. Negotiating the establishment of those *peer-to-peer* relationships is often a complicated process since technical and economical factors, as exposed in [62], need to be taken into account.

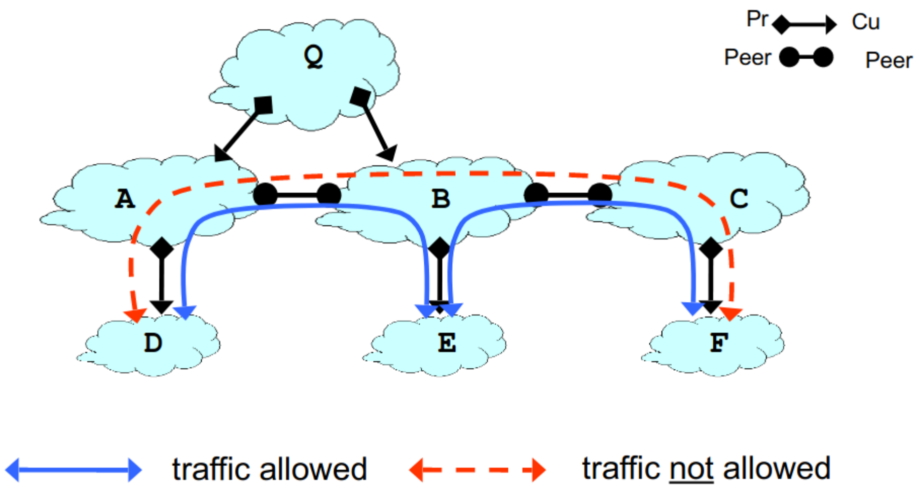


Figure 2.5: According to Gao-Rexford rules, AS D can not forward traffic towards AS F through *AS_PATH*: [A B C F]. AS E is allowed to send traffic towards AS F through *AS_PATH*: [B C F]

2.2.6 Gao-Rexford Rules

BGP is an interdomain routing protocol that allows ASes to apply local policies for selecting routes and propagating routing information, without revealing their policies or internal topology to other ASes. However, a collection of ASes may have conflicting BGP policies that lead to route divergence [112], [89]. Route divergence can result in route oscillation, which can significantly degrade the end-to-end performance of the Internet. Avoiding these conflicting BGP policies is crucial for the stability of the Internet routing infrastructure.

In [80], they propose, a set of guidelines(known as Gao-Rexford rules) that each AS should follow to set its routing policies without requiring coordination with other ASes. *Gao-Rexford guidelines ensure, BGP convergence and also, stability on the Internet routing.* The rules are listed below:

- **Exporting to a provider:** Exchanging routing information with a provider, an AS can export its routes and the routes of its customers, but can not export routes learned from other providers or peers. That is, an AS does not provide transit services for its provider.
- **Exporting to a customer:** Exchanging routing information with a customer, an AS can export its routes, as well as routes learned from its providers and peers. In that case, an AS provides transit services for its customers.
- **Exporting to a peer:** Exchanging routing information with a peer, an AS can export its routes and the routes of its customers, but can not export the routes learned from other providers or peers. This means that, an AS does not provide transit services for its peers.
- **For a destination *p***, prefer routes coming from customers over peers and peers over providers.

2.2.7 Prefix Hijacking

BGP Prefix Hijacking, also called route hijacking or IP hijacking, is the illegitimate takeover of groups of IP addresses by corrupting Internet routing tables maintained using the Border Gateway Protocol (BGP). This phenomenon can be caused by router misconfiguration (e.g., from network operators) or malicious attacks (e.g., traffic redirection, DDoS). There are three basic prefix hijacking techniques.

The first technique is the announcement of a more specific prefix than the announced prefix of a Legitimate AS from a Hijacker AS. This is commonly called ***Sub-prefix Hijacking*** or ***"sub-MOAS(i.e., sub multi origin AS)" attack***. In that case, all the traffic for that (specific) prefix is redirected to Hijacker AS, because always in routing the longest prefix match is preferred (see section 2.2.4).

The second technique is the announcement of exactly the same prefix (called as ***Prefix Hijacking*** or ***"MOAS hijack"***) as the announced prefix of Legitimate AS from Hijacker AS, but now the number of AS hops between Hijacker AS and

AS which produces the traffic for this prefix is less than the Legitimate AS. In that case, Hijacker AS is preferred due to shortest AS PATH. However, in reality, routing policies that prevail over path lengths may come into play, making the situation reached after an exact-prefix hijacking attempt more complex to assess.

The third technique called ***AS_ PATH forgery or Man in the Middle attack***. In this attack, the hijacker may arbitrarily tamper with the AS path in BGP update messages [10]. Instead of forging the origin AS, he modifies the AS path to avoid a MOAS conflict and causes one-hop (Type-1) or N-hop (Type N) prefix hijack (i.e., the hijacker AS is N hops away from the victim AS in the manipulated *AS_ PATH*). For this reason, the attacker announces the manipulated *AS_ PATH* to its neighboring ASes. Another version of this attack is the modification of the *AS_ PATH* and the advertisement of more attractive (e.g., shorter) routes at the control plane [20], but still the usage of another sequence of ASes at the data plane [24] from the hijacker AS to forward the traffic. This attack is known as traffic attraction attack and is usually performed due to economic incentives.

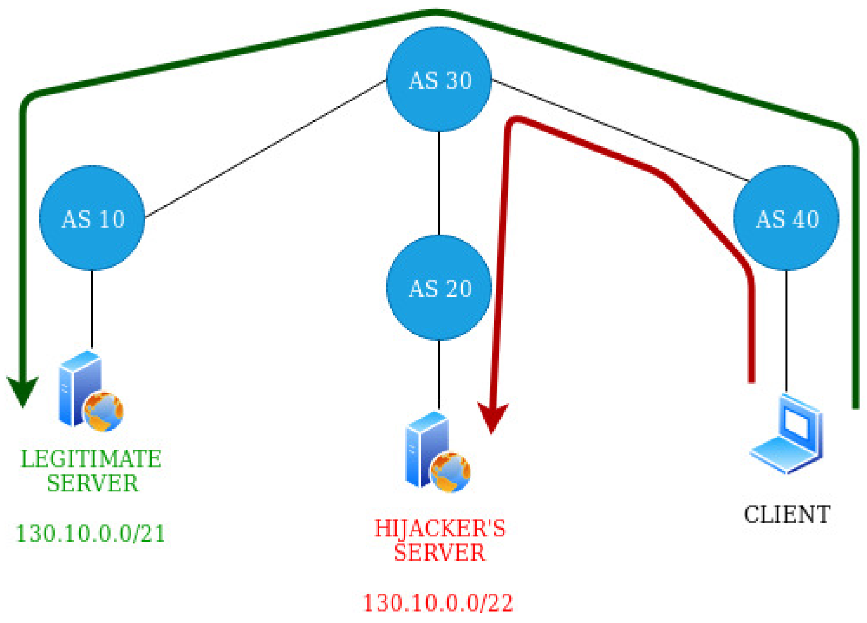


Figure 2.6: Sub-prefix hijacking example. AS 20 steals all the traffic destined to AS 10, due to more specific announcement.

In Figure 2.6, we see a Sub-prefix Hijacking example. Legitimate AS 10 announces the prefix 130.10.0.0/21 via BGP which is used by an email server. A client into AS 40 produces traffic to AS 10 in order to access the email server. At this point all the produced traffic from the client goes to AS 10. Malicious AS 20 decides to hijack all this traffic between AS 10 and AS 40 and announces two /22 sub-prefixes 130.10.0.0/22 and 130.10.4.0/22. From that moment on all the traffic is redirected to hijacker AS, because in routing the longest prefix match wins.

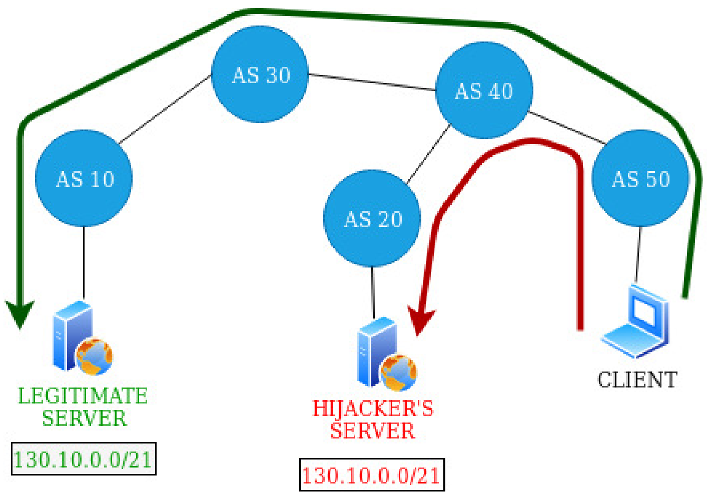


Figure 2.7: Same-prefix hijacking example. AS 20 steals all the traffic destined to AS 10, due to shortest *AS_ PATH*.

In Figure 2.7, we see a Same-prefix Hijacking example. Legitimate AS 10 announces prefix 130.10.0.0/21 via BGP which is used by a web server. A client is located in AS 50 and produces traffic to AS 10 in order to access the web server. At this point all the produced traffic from the client goes to AS 10. Similar to the above example malicious AS 20 decides to hijack all this traffic between AS 10 and AS 50. For this reason it announces the prefix 130.10.0.0/21. As we can see the number of AS hops between AS 50 and AS 20 (2 hops away) is less than the number of AS hops between AS 50 and AS 10 (3 hops away). Also, we have an announcement of the same prefix by two different ASes. So in that case BGP takes into account the shortest policy compliant *AS_ PATH* for routing.

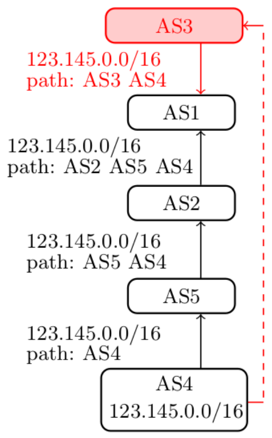


Figure 2.8: One hop prefix hijack (Type-1) [110]. The attacker (AS 3) announces a fake link between his AS and the victim AS to its neighbor (AS 1).

In figure 2.8, we see an *AS_PATH* forgery attack. AS 4 (victim) announces the prefix 123.145.0.0/16 in the Internet, so all the ASes in the topology have a valid *AS_PATH* towards AS 4. The AS 3 (attacker) decides to advertise to AS 1, a BGP route including the fake *AS_PATH*: [3 4] for the prefix 123.145.0.0/16 . This announcement constitutes a Type-1 hijack which infects only the AS 1, because AS 1 prefers the shortest route, in terms of number of AS hops ([2 5 4] $>$ [3 4]), to forward the traffic destined to AS 4.

2.3 Resource Public Key Infrastructure (RPKI)

The Resource Public Key Infrastructure (RPKI) is a hierarchical certification system that associates public keys [51] with network resources such as IP prefixes [103]. After certifying their IP prefixes, owners can use their private keys to authorize specific AS numbers to advertise these prefixes. Authorizations are cryptographically signed and published in public repositories, which enables other ASes to verify the authorization and filter routes with invalid origins to protect against prefix hijacks. The RPKI certification and validation system consists of:

- **Resource Certificates (RC):** certificates for ownership of IP prefixes (and ASNs) mapping owner public keys to IP prefixes (and ASNs).
- **Route Origin Authorizations (ROAs):** signed statements using the certified private key of the owner of an IP prefix to specify an AS number to be authorized to originate this prefix in BGP. The ROA might also permit the AS to advertise sub-prefixes, up to a specified maximum prefix length.
- **Route Origin Validation (ROV):** filtering rules to be applied by BGP routers to discard or depreference a BGP advertisement whose origin AS does not match with the information in the prefix's ROA.

2.3.1 Certification - Authorization Process

RPKI Resource Certification (RCs) is a hierarchical process with predefined steps that must be followed by the involved entities. At the top of the hierarchy we have the five Regional Internet Registries (RIRs) [56]: ARIN, APNIC, AFRINIC, LACNIC, and RIPE. Each RIR holds a root (self-signed) RC covering all the IP addresses in its geographical region. When an IP prefix is directly allocated to an organization by a RIR it can request from the RIR to issue an RC, validating its ownership of the IP prefix.

For example, OTEGlobe in Figure 2.9 was certified by RIPE for its address space 62.75.1.0/24 and is enabled to use its RC to issue a ROA protecting its IP prefix against hijacking. In case that the ownership of an IP prefix is later delegated, that is, if an organization A further allocated a subprefix to organization B, then A is responsible for certifying B as the owner for that subprefix. To accomplish this, A must itself possess an RC for its assigned IP addresses.

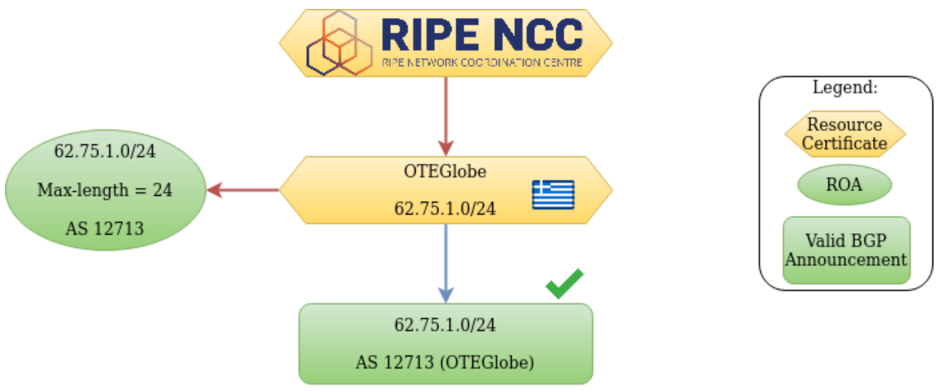


Figure 2.9: OTEGlobe received an RC from RIPE and issued a ROA to protect its IP-prefix (advertised through BGP)

2.3.2 Route Origin Validation Process (ROV)

Route Origin Validation (ROV), defined in RFC 6483 [95], allows BGP routers to prevent prefix hijacking by detecting that an incoming BGP advertisement is inconsistent with ROAs in RPKI. Large vendors support ROV in their BGP routers with negligible computational overhead [19], [35].

The first step of the ROV process (see Figure 2.10), is the periodically synchronization of the ROV enabled router with the cached RPKI databases of the five RIRs, to retrieve the updated ROAs. Upon receiving a BGP route-advertisement, the BGP router checks whether the advertised destination IP prefix p is "covered" by a ROA, that is, whether there exists a ROA for a superprefix P $\supseteq p$. The route-advertisement is then assigned by one of the following three states:

- **Not-Found:** p is not covered by any ROA.
- **Valid:** p is covered by a ROA, the origin AS number matches the AS number specified in the ROA and p is not more specific than is allowed by the maximum length specified in the ROA (e.g., see OTEGlobe's advertisement in Figure 2.9).
- **Invalid:** Otherwise (p is covered, but not "valid").

Routers use those three states to perform route-filtering policies. The default action for most of the routers is to discard invalid routes (see [19], [35]) and this is also considered as the best practice (according to RFC7454, routers SHOULD discard invalid routes). However, a ROV-enforcing AS may instead choose to configure its router to prefer invalid routes over other routes. However, as observed in [74], [92], preferring invalid routes leaves the AS completely vulnerable to subprefix hijacking. We mention that the certification and validation process in RPKI may slightly differ from AS to AS and RIR to RIR, but the main idea is the same as described in this section and section 2.3.1 .

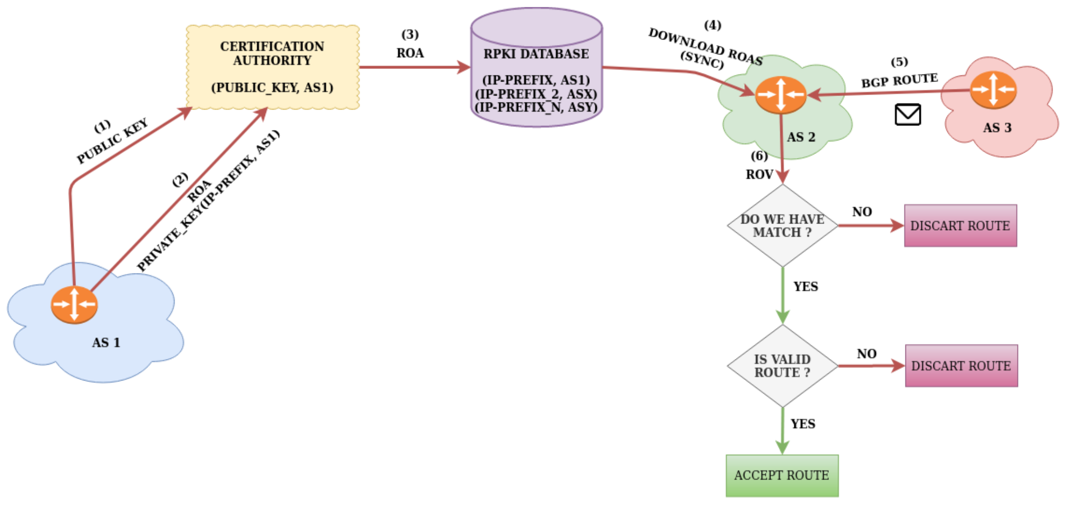


Figure 2.10: RPKI Certification and Validation Process step by step.

2.3.3 RPKI Relying Parties (RPs)

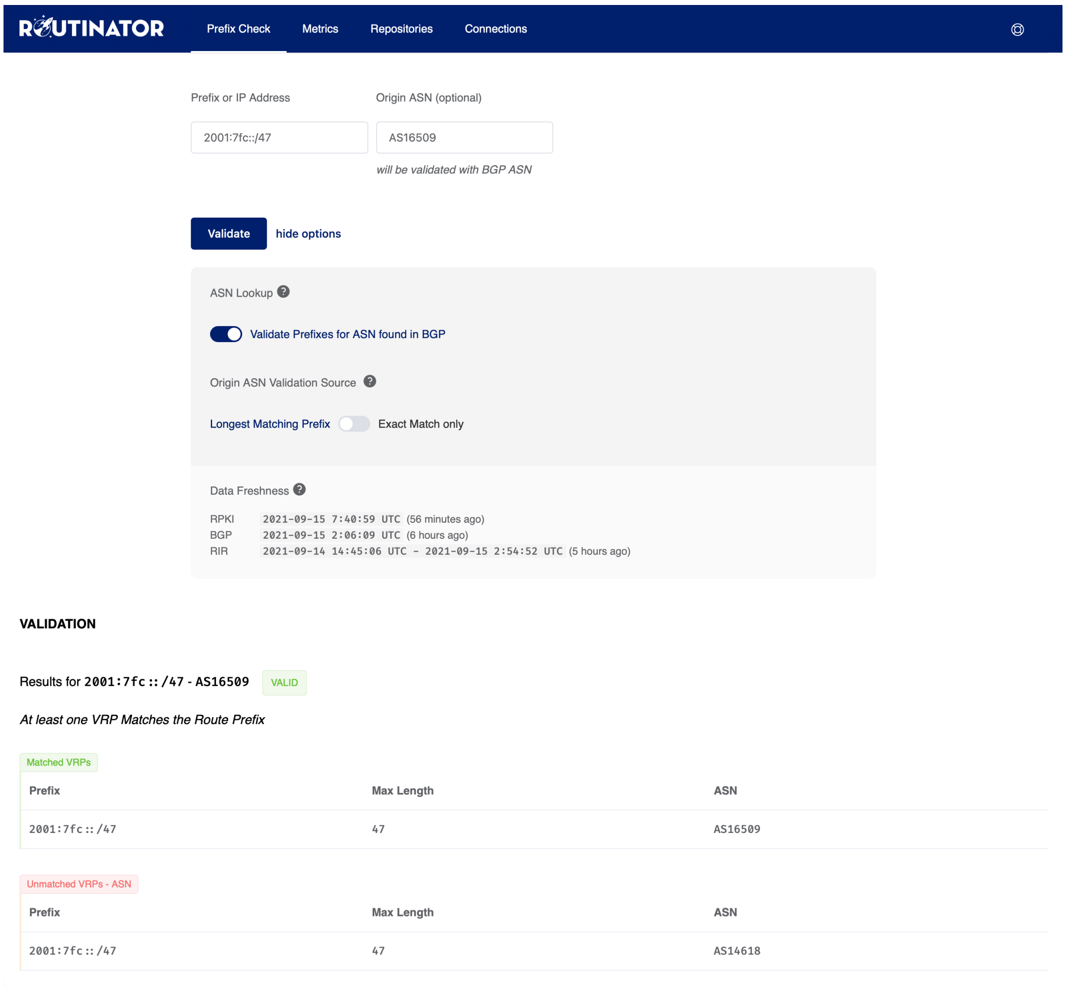
All RPKI objects (e.g., route origin authorization (ROA)) are disseminated as files in a distributed repository of *publication point (PP)* servers. Analogous to a DNS authoritative server and DNS resolvers, a PP makes RPKI data available to *relying parties (RPs)*. In contrast to DNS resolvers, which fetch data on demand and have a partial view of the DNS, these RPs must periodically fetch all authoritative data and maintain a complete view. RPs use rsync [52] or RRDP [66] for data retrieval, then cryptographically validate received RPKI objects, cache the results, and relay data such as valid prefix-to-origin AS bindings to BGP routers for use in the route decision-making process (see Figure 1 in [101]). In the next subsection, we present one of the most commonly used RPKI Relying Parties, called "Routinator".

2.3.3.1 Routinator

Routinator 3000 [49], is a free open-source RPKI Relying Party software written by NLnet Labs in the Rust programming language. The application is designed to be secure and has great portability. It is a lightweight implementation that can run effortlessly on almost any operating system using minimalist hardware.

Routinator connects to the Trust Anchors of the five Regional Internet Registries (RIRs) — APNIC, AFRINIC, ARIN, LACNIC and RIPE NCC — downloads all of the certificates and ROAs in the various repositories, verifies the signatures and makes the result available for use in the BGP workflow.

It is a full featured software package that can perform RPKI validation as a one-time operation and store the result on disk in formats such as CSV and JSON, or run as a service that periodically downloads and verifies RPKI data. Routers can connect to Routinator to fetch verified data via the RPKI-RTR protocol [53]. The built-in HTTP server offers a user interface and endpoints for the various file formats, as well as logging, status and Prometheus monitoring.



Prefix or IP Address: 2001:7fc::/47

Origin ASN (optional): AS16509

will be validated with BGP ASN

ASN Lookup

Validate Prefixes for ASN found in BGP

Origin ASN Validation Source

Longest Matching Prefix / Exact Match only

Data Freshness

```
RPKI   2021-09-15 7:40:59 UTC (56 minutes ago)
BGP    2021-09-15 2:06:09 UTC (6 hours ago)
RIR    2021-09-14 14:45:06 UTC - 2021-09-15 2:54:52 UTC (5 hours ago)
```

VALIDATION

Results for 2001:7fc::/47 - AS16509 VALID

At least one VRP Matches the Route Prefix

Matched VRPs

Prefix	Max Length	ASN
2001:7fc::/47	47	AS16509

Unmatched VRPs - ASN

Prefix	Max Length	ASN
2001:7fc::/47	47	AS14618

Figure 2.11: Routinator RPKI RP Graphical User Interface.

Also, Routinator offers a user friendly GUI (see Figure 2.11) allowing users to validate prefixes against ASNs found in BGP announcements. Next to that it allows users to lookup related prefixes for the prefix they're searching for. These related prefixes can be more- or less-specific prefixes, routed in BGP or prefixes that are allocated by one of the five Regional Internet Registries.

For larger networks, the Routinator's developers offer an extra service, named "RTRTR" [54], as a companion to Routinator. This makes it possible to centralize validation performed by Routinator and have RTRTR running in various locations around the world to which routers can connect.

2.3.4 RPKI Adoption Challenges

Despite the importance of RPKI for Internet security and the extensive efforts to push its deployment forward, **RPKI adoption is very slow** (i.e., around 20% of the total ASes on the Internet) [71]. The majority of prefixes advertised in BGP are still not in the system [40] (including most IP addresses for popular web-services [126]), and few ASes filter BGP advertisements based on the information recorded in RPKI [73], [94], [116], although there is significant progress in both these fronts [70], [71]. Next, we present the basic obstacles to RPKI's ubiquitous adoption according to the literature [96], [84], [73], [93]:

- **Certification is not easy:** RPKI's certification process is manual and hierarchical. Network operators first need to request their provider, the entity that allocated the IP address space to them, to issue them a resource certificate. Since many organizations do not yet have resource certificates for their IP address blocks, in many cases this requires providers to first be certified themselves before they can issue certificates for their customers [83].

- **Human error is common:** Issuing ROAs requires network operators to manually authorize origin ASes and to specify the maximum permissible length for advertised subprefixes. However, an operator might inadvertently not authorize all origins, or restrict the maximum length to be too short, and so advertise BGP prefixes that violate their ROAs. About 10% of the BGP announcements that are covered by ROAs are invalid [71], with the vast majority of these attributed to human error. ASes that perform ROV would unwittingly discard legitimate BGP advertisements for those prefixes, and thus disconnect from legitimate destinations. This is the most common reason that many ISP's not filtering invalid routes.

- **Circular dependency:** ASes do not gain sufficient security benefits from ROV because (a) many IP prefixes advertised in BGP are not certified through RPKI, and (b) few ASes perform ROV resulting in little incentive to certify ownership over an IP prefix (issue a ROA).

2.4 Center for Applied Internet Data Analysis (CAIDA)

The Center for Applied Internet Data Analysis (CAIDA) [14], conducts network research and builds research infrastructure to support large-scale data collection, curation, and data distribution to the scientific research community. The group, located at the federally funded San Diego Supercomputer Center located at the University of California, San Diego, designs, deploys and maintains a growing number of computational, data analysis and visualization services. The group also ships and maintains small form factor measurement instrumentation to networks around the world, extending its Archipelago (Ark) Internet measurement platform [4] for use by the network and cybersecurity research community. CAIDA researchers develop novel techniques to collect, analyze, query and visualize the resulting data.

The CAIDA's mission is to investigate practical and theoretical aspects of the Internet, focusing on activities that:

- provide insight into the macroscopic function of Internet infrastructure, behavior, usage and evolution.
- foster a collaborative environment in which data can be acquired, analyzed, and (as appropriate) shared.
- improve the integrity of the field of Internet science.
- inform science, technology and communications public policies.

2.4.1 AS Relationships dataset

CAIDA collects several types of data, and makes this data available to the research community while preserving the privacy of individuals and organizations who donate data or network access.

One of the most useful datasets that CAIDA offers is the *"Inferred AS Relationships Dataset"*, which split-ted into 2 directories, called *"serial-1"* and *"serial-2"*. The *"serial-1"* directory contains AS relationships inferred from BGP using the method described in "AS Relationships, Customer Cones, and Validation" [105]. *Serial-2* adds links inferred from BGP communities [18] using the method described in "Inferring Multilateral Peering" [87] and traceroute.

Serial-1

Serial-1 data is available from 2004 to present, with one file created per week in 2006 and one per month in prior years. Each file contains a full AS graph derived from RouteViews [48] BGP table snapshots taken at 8-hour intervals over a 5-day period. The AS relationships available are customer-provider (and provider-customer in the opposite direction), peer-to-peer, and sibling-to-sibling. The general serial-1 procedure for creating a file is as follows:

1. Extract all AS links from RouteViews [48] snapshots.
2. Infer customer-provider relationships, and annotate AS links.
3. Infer peer-to-peer relationships, and annotate AS links, possibly overriding customer-provider relationships inferred in step 2.
4. Heuristically fix suspicious looking inferred relationships (e.g., a low-degree AS acting as provider to a high-degree AS).

5. Infer sibling ASes (that is, ASes belonging to the same organization) from WHOIS [59], and annotate AS links, possibly overriding previous relationship annotations.

Serial-2

Serial-2 data is available from October 2015 to the present, with one file created per week. In addition to the links from the serial-1 graph, there are AS links inferred from BGP communities [18] collected from IX looking glass servers [31] collected in a single day and tracerouter data collected on the same day from CAIDA's ark monitors. We mention that our BGP simulator uses the serial-2 dataset to model the Internet graph (i.e., ***is the core component that enabled us to develop BPHS***).The general serial-2 procedure for creating a file is as follows:

1. Collect BGP communities from IX looking glass servers.
2. Infer peering links between pairs of AS which accept routes from each other.
3. Collect archived BGP data from Routeviews and RIPE Routing Information Service (RIS) [47].
4. Infer peering links at points in the observed AS paths that cross an known Internet Exchange Point [26] .
5. Collect traceroutes from ark monitors.
6. Convert the IP path to AS path using inferred ownership and keep the first AS link in the path.
7. Merge all newly inferred links to the serial-1 graph as peering links.

2.4.2 ASRank

ASRank is CAIDA's ranking of Autonomous Systems (AS) (which approximately map to Internet Service Providers) and organizations (Orgs) (which are a collection of one or more ASes). This ranking is derived from topological data collected by CAIDA's Archipelago Measurement Infrastructure and Border Gateway Protocol (BGP) routing data collected by the Route Views Project and RIPE NCC. ASes and Orgs are ranked by their ***"customer cone size"*** (i.e., the number of their direct and indirect customers), which in turn is inferred from BGP paths by CAIDA's AS relationships inference algorithm that mentioned in the previous section.

AS customer cone

Looking specifically at the AS customer cone, we define an AS-A's AS customer cone as the AS-A itself plus all the ASes that can be reached from A following only peer-to-customer links in BGP paths we observed. In other words, A's customer cone contains A, plus A's customers, plus its customers' customers, and so on. The size of the customer cone of an AS reflects the number of other elements (ASes, IPv4 prefixes, or IPv4 addresses) found in it's set. An AS in the customer cone is assumed to pay, directly or indirectly, for transit, and provides a coarse metric of the size or influence of an AS in the routing system.

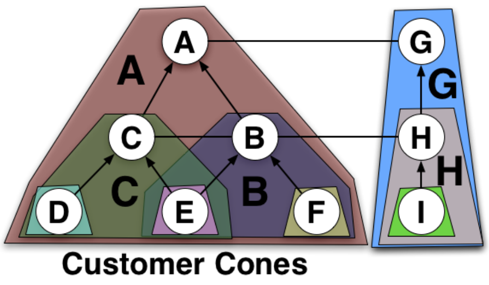


Figure 2.12: The customer cone size of each AS depicted with a unique color [11].

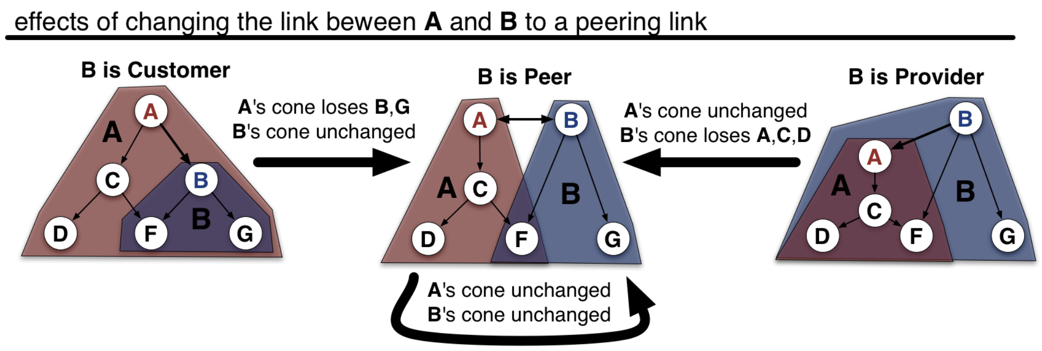


Figure 2.13: The effect on customer cones of changing the relationship of A to B from its current one to a peering relationship [11].

For example, Figure 2.12 depicts several AS customer cones, ASes D, E, F , and I all sit at the bottom of the hierarchy and so only have a single AS in their cone. H ranks a little bit higher with 2 ASes. Both C and B tie with 3 ASes. Note that B and C both have E in their respective cones. A is ranked at the top of the hierarchy with 6 ASes in its customer cone.

In Figure 2.13, we see an example illustrating how different relationships affect the customer cone sizes of AS A and B. If the original graph had B as a customer of A then A's cone contains 7 ASes: {A,B,C,D,E,F,G}. B's cone contains three ASes: {B,F,G}. If the link between A and B is changed to a peering link, A loses customers B and G, which it had access to exclusively through its customer relationship with B. A's cone does not lose F, since it can still reach it through its customer relationship with C. A's cone size thus shrinks to 4 ASes: {A,C,D,F}. Since AS B did not previously reach any customers through A, its customer cone is unaffected by this change.

2.4.3 BGP Hijacks Observatory

The BGP Hijacks Observatory is a CAIDA project to detect and characterize BGP hijacking attacks, including stealthy man-in-the-middle (MiTM) Internet traffic interception attacks. Observatory uses the HI^3 PaaS [12] to power its data collection and analytics platform, and provides event data to HI^3 to allow correlation with other types of Internet security data. The Observatory serves multiple purposes:

- A platform for operators to troubleshoot anomalous events and enable situational awareness.
- Research on Hijacks and BGP anomalies.
- A Testbed to realistically experiment with detection techniques applied to Internet routing data in the wild.
- A Testbed for developing new inference methods.

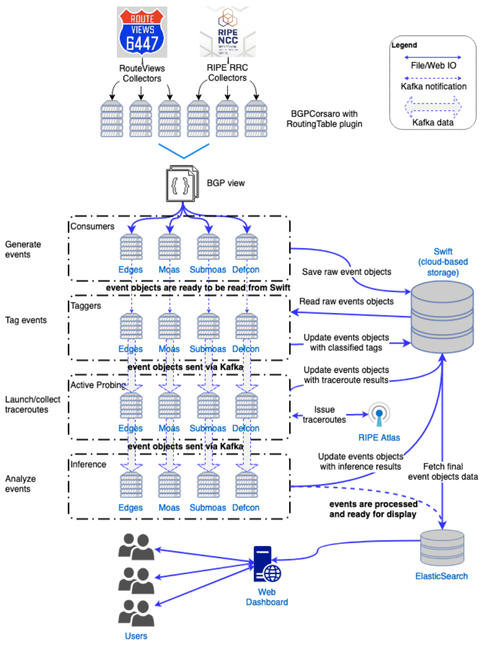

ROUTE VIEWS 6447
RIPE NCC
RouteViews Collectors
RIPE RRC Collectors
Legend
File/Web IO
Kafka notification
Kafka data
BGPCorsaro with RoutingTable plugin
BGP view
Consumers
Generate events
Edges
Moas
Submoas
Defcon
Save raw event objects
Swift (cloud-based storage)
event objects are ready to be read from Swift
Taggers
Tag events
Read raw events objects
Update events objects with classified tags
event objects sent via Kafka
Active Probing
Launch/collect traceroutes
Update events objects with traceroute results
Issue traceroutes
RIPE Atlas
event objects sent via Kafka
Inference
Analyze events
Update events objects with inference results
Fetch final event objects data
events are processed and ready for display
ElasticSearch
Web Dashboard
Users

Figure 2.14: CAIDA Hijacks Observatory, architecture [13].

The Hijacking Observatory (see Figure 2.14), monitors (24/7) the state of BGP routing using RIPE and RouteViews data to detect anomalies that may be events of BGP hijacking. For each suspicious event, the system augments control-plane (BGP) data with data-plane measurements (traceroutes) executed from a set of RIPE Atlas probes [46], on-the-fly. Events are enriched with descriptive tags based on various database lookups (e.g., AS relationships, AS Customer Cone, IXP prefixes) and heuristics (e.g., potential “fat finger” misconfiguration) and presented to users through a Web interface that facilitates inspection of events.

2.5 Basic concepts in Network Simulation

In this section, we introduce some basic concepts in the area of network simulation. We firstly introduce the basic idea of "network simulation" in computer science then the different types of network simulators that exist and finally, we discuss the difference between simulation and emulation.

2.5.1 Simulation

The imitation of the real-world's conditions and processes in the course of time is known as ***simulation****.* By simulation, the system behavior can be characterized and analyzed, what-if questions can be raised, and systems with close similarity to real conditions can be designed. Significant information regarding the feasibility, productivity, and efficiency of a system can be assessed by simulation prior to real deployment of actual implementation [61].

Three types of simulation have been mentioned in computer science literature [121], a) *Monte Carlo simulation*, b) *Trace-driven simulation*, and c) *Discrete-event simulation*:

- **Monte Carlo simulation**, is a static simulation or one without a time axis. It is used for modeling probabilistic events whose characteristics do not vary over time. Also, Monte Carlo simulation is utilized to appraise non-probabilistic expressions by making use of probabilistic approaches.

- **Trace-driven simulation**, uses a trace as an input in the process of simulation. A trace is defined as a time-ordered history of phenomena in a real system. In general, Trace-driven simulation is used in analyzing or tuning resource management algorithms.

- **Discrete-event simulation**, in contrast to continuous-event simulation, uses a discrete-state model of the system for simulation and is used due to the variable system state which is described by the number of jobs at various devices. Time in discrete-event simulation can be discrete or continuous [97].

2.5.2 Why Network Simulation?

Network simulation, is a technique of implementing a network on a virtual environment (e.g., a computer). Through this, the behavior of the network is calculated either by network entities interconnection using mathematical formulas or by capturing and replaying observations from a production network [123].

In the network research area, *it is very costly (in terms of computation resources) to deploy a complete testbed containing multiple networked computers, routers and data links (e.g., the Internet graph) to validate and verify a certain network protocol or a specific network algorithm.* The network simulators in these circumstances save a lot of money and time in accomplishing this task. Network simulators are also particularly useful in allowing the network designers to test new networking protocols or to change the existing protocols in a controlled and reproducible manner. One can design different network topologies using various types of nodes (e.g., hosts, hubs, bridges, routers and mobile units). Afterwards, the routing behavior can be easily studied in different topologies, given the fact that the network topology is merely a set of simulation parameters.

2.5.3 Types of Network Simulators

Currently there is a large variety of network simulators, ranging from the simple ones to the complex ones. Minimally, a network simulator should enable users to represent a network topology, defining the scenarios, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to process network traffic.

There are different types of network simulators which can be classified into three basic categories (according to [90]):

- **Graphical Simulators:** allow users to easily visualize the workings of their simulated environment (see Figure 2.15).
- **Text-based Simulators:** provide a less visual or intuitive interface, but may allow more advanced forms of customization.
- **Programming-Oriented Simulators:** provide a programming framework that allows users to customize an application that simulates the networking environment for testing.

2.5.4 Advantages and Drawbacks

Below we list the main advantages and drawbacks of simulation tools and techniques, according to [90]:

Advantages

- Sometimes cheaper than real testbeds.
- Find bugs (in design) in advance.
- Generality: over analytic/numerical techniques.
- Detail: can simulate system details at an arbitrary level.

Drawbacks

- Caution: does the simulation model reflects reality?

- Large scale systems: lots of resources to simulate (especially accurately simulate).
- May be slow (computationally expensive – 1 min real time could be hours of simulated time).
- Art: determining the right level of model complexity.
- Statistical uncertainty in results.

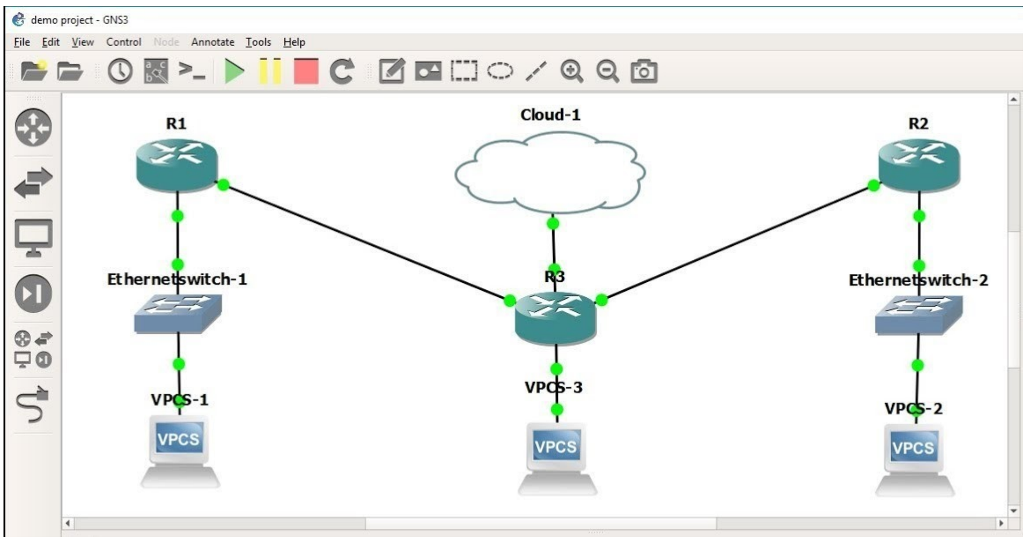


Figure 2.15: GNS3 virtual lab environment. The network topology consists of 3 routers, 2 Ethernet switches, 3 VPCS and a Cloud node for internet access.

2.5.5 Network Simulation versus Emulation

Network simulation, is a useful technique since the behavior of a network can be modeled by calculating the interaction between the different network components (they can be end-host or network entities such as routers, physical links or packets) using mathematical formulas. They can also be modeled by actually or virtually capturing and playing back experimental observations from a real production networks. After we get the observation data from simulation experiments, the behavior of the network and protocols supported can then be observed and analyzed in a series of offline test experiments. All kinds of environmental attributes can also be modified in a controlled way to assess how the network can behave under different parameter combinations or different configuration conditions. Another characteristic of network simulation that is worth noticing is that the simulation program can be used together with different applications and services in order to observe end-to-end or other point-to-point performance in the networks.

Network emulation, however, means that the network is simulated in order to assess its performance or to predict the impact of possible changes, or optimizations. The major differences between them is that a network emulator means that end-systems such as computers can be attached to the emulator and will act exactly as they are attached to a real network. The major point is that the network emulator's job is to emulate the network which connects end-hosts, but not the end-hosts themselves.

Typical network emulation tools include Graphical Network Simulator 3 (GNS3) [27], which is a popular network emulator that can also be used as a simulator. Specifically, GNS3 (see Figure 2.15), is a network software emulator that allows the combination of virtual and real devices, used to emulate complex networks. GNS3 is open-source and supports many networking vendors like Cisco, Juniper, Arista and others. GNS3 also allows us to add virtual machines of almost any type in our GNS3 topologies. GNS3 is used by many large companies around the world for testing networks before they are put into production. In contrast, Network Simulator 3 (NS3) [38] is a typical discrete-event network simulator for Internet systems, targeted primarily for research and educational use.

2.6 Full Stack Web Application Development

A **web application (or web app)** is an application software that runs on a web server, unlike computer-based software programs that run locally in the operating system (OS) of the device. Web applications are accessed by the user through a web browser with an active network connection. These applications have been programmed using a *client–server* modeled structure — the user ("client") is served services through an off-site server that is hosted by a third-party.

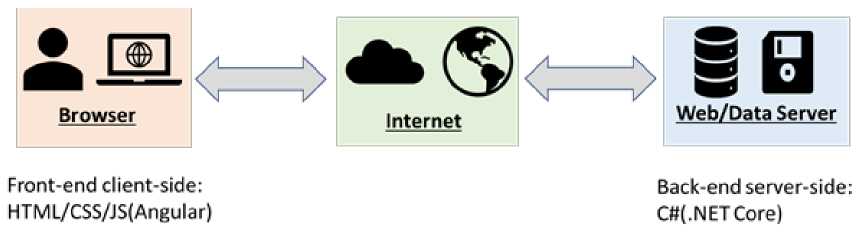


Figure 2.16: Full Stack Web Development [104].

Web application development normally involves both client-side and server-side programs, which is often referred to as full-stack web development. Among them, client-side is also called *front-end* and server-side is called *back-end*. As shown in Figure 2.16, the front-end program, most of times, is written in HTML/CSS/Javascript, and in the last few years Angular, Vue and React have emerged as front runner frameworks. For back-end development, it can be programmed with Java, Python, C#, node.js and PHP, and data server can either be *SQL based* such as MySQL, PostgreSQL or *NoSQL based* [122] such as MongoDB.

2.6.1 Model-View-Controller (MVC) Pattern

Model–view–controller (MVC) is a software design pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. Traditionally used for desktop graphical user interfaces (GUIs), this pattern became popular for designing web applications. Popular programming languages have MVC frameworks that facilitate implementation of the pattern.

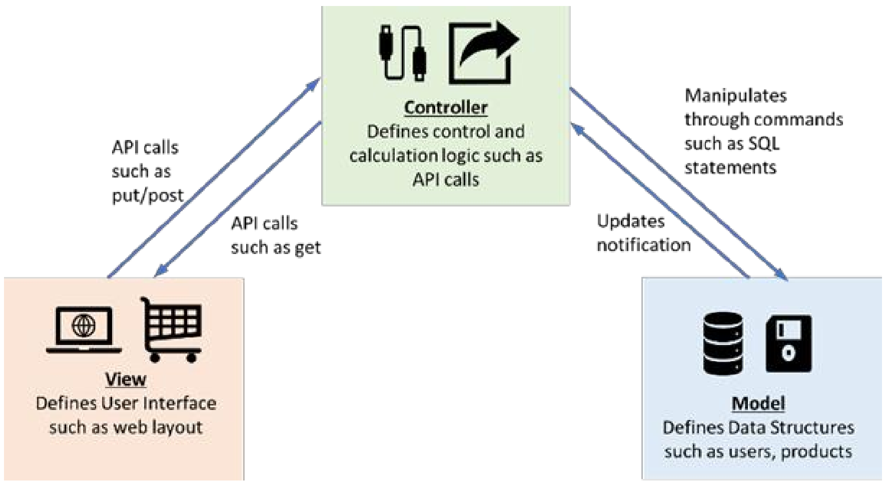

Controller
Defines control and calculation logic such as API calls
API calls such as put/post
API calls such as get
Manipulates through commands such as SQL statements
Updates notification
View
Defines User Interface such as web layout
Model
Defines Data Structures such as users, products

Figure 2.17: Model-View-Controller Design [104].

As shown in figure 2.17, there are three components in MVC, namely model, view and controller:

- **Model:** The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the web application.
- **View:** Any representation of information such as a chart, diagram or table. It defines the user interface such as web layout and doesn't know the application's Model neither directly interacts with it.
- **Controller:** serves a bridge between model and view, it processes application requests from view and then forwards them to model and also carries updates from model to view.

In MVC each component can maintain its independence, which offers flexibilities for the developer to select the most suitable frameworks/programming languages for the front-end and the back-end. For example, one of the popular picks in industry is Angular or ReactJS front-end plus ASP.NET core framework using C# programming language for back-end.

2.6.2 RESTful API

API stands for application programming interface; it is an interface for the website or mobile application to communicate with the backend logic. For example in Figure 2.17 View communicates with Controller through a common API, to request or send the appropriate data. In other words it is like a messenger that takes a request from users and sends it to the backend system. Once the backend system responds, it will then pass that response to the users. The most common API in the Web programming is called *"REST"*.

REST stands for *Representational State Transfer.* It is considered the bible in the web domain, however, it is not a standard or protocol; it is more like a software architectural style. Many engineers follow this architectural style to build their application, such as eBay, Facebook, and Google Maps. These web applications serve huge amounts of traffic every second, so we can infer that REST is a scalable architectural style. There are five important constraints/principles for the REST architecture style [68]:

- **Client-Server:** There is an interface between the client and server. The client and server communicate through this interface and are independent of each other. Either side can be replaced as long as the interface stays the same. Requests always come from the client-side.
- **Stateless:** There is no state for the request. Every request is considered independent and complete. There is no dependence on the previous request nor dependence on a session to maintain the connection status.
- **Cacheable:** Things are cacheable on the server or client-side to improve performance.
- **Layered system:** There can be multiple layers in the system, and the goal here is to hide the actual logic/resources. These layers can perform different functions, such as caching and encryption.
- **Uniform interface:** The interface stays the same. This helps to decouple the client and server logic.

2.6.3 HTTP Protocol

HTTP stands for HyperText Transfer Protocol; and it is an implementation of the REST architecture style that we described in the previous section. *HTTP is the standard protocol used on the World Wide Web (WWW).* We use it every day to browse different websites.

In the HTTP protocol, there are different types of service request methods. Each service request method has a special definition that is specific to it. When the frontend interface interacts with the backend API through a URL, it also needs to define the HTTP method for this request. For example, reading and creating data are completely different services, so they should be handled by different HTTP methods. There are five basic HTTP methods:

- **GET:** For reading data.
- **POST:** For creating data.
- **PUT:** For updating data by completely replacing data with new content.
- **PATCH:** For updating data, but by partially modifying a few attributes.
- **DELETE:** For deleting data.

HTTP Status Codes

An HTTP status code is a code that is returned in the HTTP protocol. The status code helps the frontend client understand the status of their request, that is, whether it is success or failure. Next we list some of the most commonly used status codes:

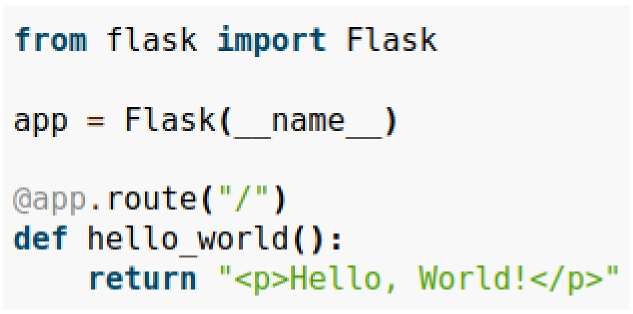
- **200 OK:** the request was successful. The request could be a GET, PUT or PATCH.
- **201 CREATED:** the POST request was successful and a record has been created.
- **204 NO CONTENT:** the DELETE request was successful.
- **400 BAD REQUEST:** the request is incorrect or corrupted and the server couldn't understand it.
- **401 UNAUTHORIZED:** the client request is missing authentication details.
- **403 FORBIDDEN:** the requested resource is forbidden.
- **404 NOT FOUND:** the requested resource does not exist.
- **500 INTERNAL SERVER ERROR:** indicates that the server encountered an unexpected condition that prevented it from fulfilling the request.

2.6.4 The Flask Web Framework

A web framework is an architecture containing tools, libraries, and functionalities suitable to build and maintain massive web projects using a fast and efficient approach. They are designed to streamline programs and promote code reuse. To create the server-side of the web application, you need to use a server-side language. Python is frequently used to such frameworks, famous among which are Django and Flask.

Python Flask Framework [23], is a lightweight micro-framework based on Jinja2 [34], Werkzeug (WSGI) [58]. It is called micro framework because it aims to keep its core functionality small yet typically extensible to cover an array of small and large applications. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Even though we have a plethora of web apps at our disposal, Flask tends to be better suited due to:

- Built-in development server, fast debugger.
- Integrated support for unit testing.
- RESTful request dispatching.
- Jinja2 Templating.
- Support for secure cookies.
- Lightweight and modular design allows for a flexible framework.



```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

Figure 2.18: A basic Flask application.

In Figure 2.18, we see an example of a basic Flask application that proves the simplicity to instantiate a Flask app. Next, we describe the code snippet:

1. First we imported the *"Flask"* class. An instance of this class will be our WSGI application.
2. Next we create an instance of this class. The first argument is the name of the application's module or package. *__name__* is a convenient shortcut for this that is appropriate for most cases. This is needed so that Flask knows where to look for resources such as templates and static files.
3. We then use the *route()* decorator to tell Flask what URL should trigger our function.

4. The function returns the message we want to display in the user's browser. The default content type is HTML, so HTML in the string will be rendered by the browser.

2.6.5 ReactJS

ReactJS [44], is a component based library which is deployed for the development of interactive user interfaces. Currently it is one of the most popular front-end JS library. It incorporates the view (V) layer in M-V-C (Model View Controller) pattern. It is supported by Facebook, Instagram and a community of individual developers and organizations. React basically enables development of large and complex web based applications which can change its data without subsequent page refreshes (*S*ingle *P*age *A*pplications). It targets to provide better user experiences and with blazing fast and robust web apps development. ReactJS can also be integrated with other JavaScript libraries or frameworks in MVC, such as AngularJS [2]. Next, we list the main features of the library:

- **Highly efficient performance**, due to the Virtual lightweight DOM (Document Object Model) of the framework.
- **Declarative**, enabling users to easily create interactive UI.
- **Component-Based**, enabling users to write reusable code.
- **Easy learning curve**.

2.6.6 PostgreSQL

PostgreSQL [43], is an advanced, enterprise class open source relational database that supports both SQL (relational) [55] and JSON (non-relational) [33] querying. It is a highly stable database management system, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity, and correctness. PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications. PostgreSQL has many advanced features that other enterprise-class database management systems offer, such as:

- User-defined types and table inheritance.
- Sophisticated locking mechanism.
- Views, rules, subquery.
- Nested transactions (savepoints).
- Multi-version concurrency control (MVCC).
- Asynchronous replication.

Chapter 3

Related Work

In the network research area, simulators help the network developers to evaluate whether a network is able to work in the real world, to identify the security issues of a networking protocol or to change the existing protocols in a controlled and reproducible way. Simulation frameworks/systems are widely accepted by the Internet-Measurements community, because commonly, the researchers want to conduct large-scale evaluation experiments for a study that is difficult and risky to be done on the real Internet (e.g., BGP prefix hijacking attacks that may affect the normal routing of the Internet, ROV adoption benefits) [72], [85], [86], [102].

In order to model the Internet graph and the interdomain-routing in a virtual environment, researchers need simulators that are highly scalable and efficient (i.e., to generate the complete AS graph [79] and simulate the interdomain-routing as quickly as possible with the least possible CPU and memory resources). For example, there are very advanced simulators like GNS3 [27] that can simulate networks by fully emulating each router (see section 2.5.5). However, doing this for the whole Internet with above 70K nodes is not practical, because this approach requires a huge amount of computing resources.

In this section, we present a brief review of the existing BGP simulators that have been developed by the network community trying to overcome the problems presented above. We briefly list the capabilities and limitations of different BGP simulators that have been used by several studies focusing on network reliability, BGP convergence, and routing security. Also, we present custom BGP simulators that have been developed by the research community to study the impact of BGP prefix hijacking on the Internet and the contribution of the RPKI mechanism.

3.1 Internet/BGP Simulators: A brief review

There are a number of BGP simulators publicly available. Although the simulators differ in certain points they show some significant resemblance: BGP is fully or partially implemented. BGP simulators can be categorized into 2 main categories: AS-level and packet-level simulators. AS-level simulators model the interdomain routing system as a graph G = (V, E) where the vertices are autonomous systems (ASes), and edges indicate physical connections between ASes. Packet-level simulators aim to fully simulate the BGP protocol as detailed as possible (e.g., at a very low level like TCP). Next, we present widely used AS-level and packet-level simulators (according to our literature research).

BGP++ [76], is an event-driven, packet-level BGP simulator using the Zebra bgpd daemon [25] on top of the ns-2 network simulator [57]. The BGP++ simulator actually simulates the network itself (in ns-2) at a very low level. BGP is not abstracted, but runs as a full-featured BGP implementation (i.e., same as it used in the real world). BGP++ runs BGP without any abstraction and therefore is not suitable for simulating large-scale topologies (i.e., the Internet), because of memory and CPU requirements.

C-BGP [113], is a BGP simulator addressing routing policy evaluation. The backbone network is not simulated in detail as in BGP++ (e.g., TCP level, connection creation, keep-alive messages were ommited), but C-BGP still uses a full BGP implementation. It is possible to see the exact state of each node (with all BGP parameters) during the simulation and each BGP decision by evaluating all policies, preferences, and tie-breaking rules. In addition, C-BGP can be used on large topologies with sizes of the same order of magnitude as the Internet (e.g., bigger than 70K AS nodes).

Karlin *et al.* [100] and Wojciechowski et al [127], propose 2 AS-level simulators that are more scalable than BGP++ and C-BGP but the computational overhead of these simulators remains high because they use discrete event simulation (see section 2.5.1).

Chaos [15], is an AS-level E-BGP, simulation framework with the capability to take as input various topological, bandwidth, adversarial (i.e. botnet), and other arbitrary models and drive experimentation with topologies in a scale of the modern Internet. Additionally, Chaos can track generic units of "traffic" across AS-to-AS links, which enables tracking congestion (useful for DDoS research). Chaos outperforms the simulation environments of GNS3, OmniNet and BGP++ that fully simulate BGP, since they suffer from lack of scalability or cost money (i.e., computational resources). C-BGP comes closest to Chaos, even if it lacks an intuitive binary interface, and is necessary to be configured via the command-line. While Chaos is considerably more scalable than previous works, according to technical specifications (that are presented in the officially repository of the framework), if a user wants to simulate the full Internet topology, it must be deployed on a supercomputer with 50+ cores and 200+ GB of RAM.

The researchers have tried to mitigate the scalability issues of the aforementioned AS-level simulators by scaling down the AS graph itself using AS-graph generation tools like GT-ITM [128], BRITE [108], or Inet [98]. The problem with these techniques is that they generate *synthetic* AS graphs. Synthetic topologies are generated from modeling the network applying mathematical algorithms, however, *non-synthetic* topologies are generated from real measurements taken from the network. The Internet's structure change dynamically and quickly (e.g., new ISPs and links are added every month), so it is not effective (i.e., inaccurate simulation results) to use synthetic topologies to simulate the Internet.

Gill *et al.* [86], was the first study that tried to avoid potential inaccuracies that might result from scaling down the AS-level topology, by running simulations on full empirical AS graphs [69], [75]. In this work, the researchers model the AS-level topology of the Internet as graph G = (V, E) where vertices represent ASes and edges represent connections between them allowing traffic exchange. Each edge is annotated with the standard model for business relationships in the Internet (see section 2.2.5): (a) customer-provider, where the customer pays the provider and (b) peer-to-peer, where two ASes agree to transit each other's traffic at no cost. They assume that each AS-X computes paths to a given destination AS-Y based on (i) a ranking on outgoing paths (i.e., Local Preference, Shortest Paths, Tie Break), and (ii) an export policy specifying the set of neighbors to which a given path should be announced (i.e., Gao-Rexford rules, see section 2.2.6). To support running repeated simulations over the full AS graph, they parallelized them across a compute cluster running DryadLINQ [77]. Their approach drastically reduces the computational overhead due to (i) the algorithmically computed BGP paths, (ii) the abstraction of the intradomain details, and (iii) the assumption that all ASes use a common routing policies model.

Cohen *et al.* [72], introduces a similar simulation framework as in [86] (i.e., CAIDA AS-level graph, Gao-Rexford rules, BGP selection path algorithm) to evaluate their proposed modest extension to RPKI, called "path-end validation", for different attack strategies and quantify the attacker's success by the fraction of ASes that is able to attract. The only difference compared to [86], is the addition of hidden peering links within IXPs, according to CAIDA Internet eXchange Points Dataset [32] (based on research work by Giotsas *et al.* [88]).

Zeng *et al.* [129], compared with [72], use the CAIDA AS relationship dataset along with the Problink dataset [99], which is mainly inferred from the BGP table snapshots collected by public measurement points at Route Views and RIPE RIS and reveals more complex relationships than the CAIDA Relationship dataset.

Hlavacek *et al.* [93], uses the same BGP route-computation framework as in [72], with the difference that they extend the simulator to mark vantage points ASes that are used by a novel system that automatically certifies de facto ownership of IP addresses, populates public repositories, and generates filtering rules for RPKI ROV, called "DISCO".

Kotronis *et al.* [120] and Sermpezis *et al.* [119], [118], use a similar simulation framework as in [72] to evaluate the impact of different types of hijacks, the performance of different monitoring services (i.e., AS nodes), and the efficiency of various mitigation methods.

Milolidakis *et al.* [109], introduces an improved and more efficient implementation of the BGP simulator used in [118], to investigate the robustness of monitor-based detection systems with respect to so-called "smart" attackers who engineer their hijacks to evade detection. The basic/new features of this simulator is the parallelization of the simulation repetitions for quicker simulation results (using a

Python module called *"mpipe"* [37]) and the support of a new type of BGP attack called *"smart hijacking"*.

Brandt *et al.* [64], [65], introduce a novel BGP simulator for large scale evaluation of attacks against Internet networks and systems (i.e., BGP prefix hijack attacks), that achieves much higher accuracy and better performance in contrast to existing simulation platforms that presented above in this section (e.g., [72], [86], [93]). The proposed simulator [42], similarly to previous works, uses the AS Relationship dataset of CAIDA, but during the breadth-first-search on the AS-graph takes into consideration also the AS relationships converting the AS-graph to a directed graph. This technique provides a significant performance improvement, because it omits impossible hops (i.e., avoids invalid paths according to routing policies). Also, the simulator focuses on simulating only specific attacks on each run and thus pre-allocates memory and structures used for the search. This approach avoids dynamic memory allocations during simulations, resulting in better performance.

3.2 Any Web-based BGP hijacking Simulator?

Web-based BGP hijacking simulators, are easier to use compared to non-web simulators as they don't require prior installation and enable network operators or users without programming skills to easily conduct their study through a user-friendly Web Interface.

According to our extensive literature research, there is not any related work proposing a Web-based, plug&play BGP hijacking simulation tool or service. All the research works mentioned in the previous section are based on command-line BGP simulators with non-trivial installation or usage.

In this work, we introduce *BPHS*, a new Web-based BGP prefix hijacking simulation tool, that extends the simulator purposed in [119] to support (a) a user-friendly GUI for easy interaction from all end-user types, (b) multi-threading execution enabling end-users to retrieve quicker results per simulation (thanks to Milolidakis *et al.* [109]), (c) a REST API [107] enabling other applications to communicate with our simulator and (d) realtime RPKI filtering using the most up-to-date data from the RPKI databases, for more realistic simulation results.

Chapter 4

BPHS

In chapter 3, we presented several BGP simulation frameworks that have been developed by network researchers to evaluate their study (e.g., prefix hijacking attacks, RPKI ROV adoption). In this chapter, we introduce our proposed BGP prefix hijacking simulation tool, called *BPHS*. First, we briefly explain the contributions of the tool and its main features. Next, we analyze the architecture of the simulator and the function of its core components (i.e., frontend, backend and database). Finally, we list the challenges that we faced during the BPHS development.

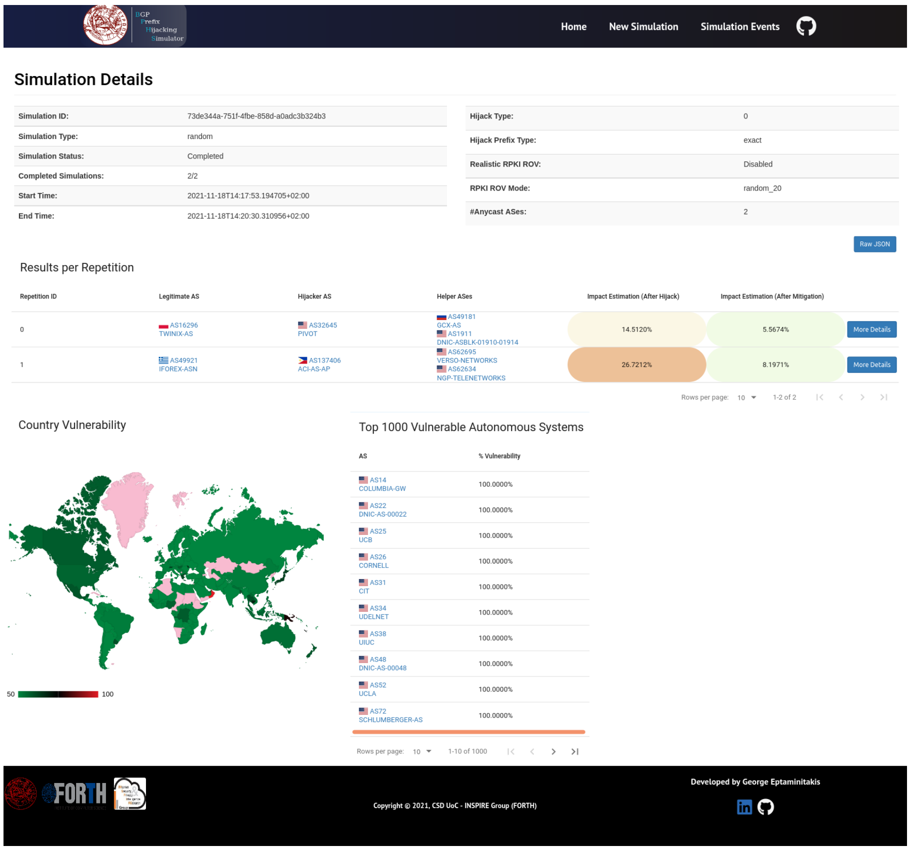
4.1 The Goal

BPHS, is the first Web-based BGP Prefix Hijacking Simulation tool that enables network operators to quickly and easily (a) assess the vulnerability of their Autonomous Systems to BGP prefix hijacks and (b) to measure the benefits of the RPKI's adoption on the Internet, through a user-friendly web application. With BPHS, the network operators can simulate all the different types of BGP hijacking attacks and obtain the simulation results through an automated and graphical way (i.e., well-designed Graphical User Interface). Also, BPHS can be offered as a Web service to the end-users, meaning that, can be publicly deployed and easily accessible by anyone in the Internet.

4.2 Supported Features

BPHS provides important features that enable end-users to fully experiment with BGP hijacking simulations, in a realistic way, and obtain meaningful results from each simulation run. Next, we list the main features of the BPHS:

- **All BGP hijacking types:** BPHS supports origin and Man in the Middle attacks (see section 2.2.7) for prefix and sub-prefix hijack attempts. For example, a user could launch the following hijacking combinations (one at a time): prefix hijack - Type 1, subprefix hijack - Type 2, prefix hijack - Type 0, or subprefix hijack - Type 0.



Simulation Details

Simulation ID:	73de344a-751f-4fbe-858d-a0adc3b324b3	Hijack Type:	0
Simulation Type:	random	Hijack Prefix Type:	exact
Simulation Status:	Completed	Realistic RPKI ROV:	Disabled
Completed Simulations:	2/2	RPKI ROV Mode:	random_20
Start Time:	2021-11-18T14:17:53.194705+02:00	#Anycast ASes:	2
End Time:	2021-11-18T14:20:30.310956+02:00		

Results per Repetition

Repetition ID	Legitimate AS	Hijacker AS	Helper ASes	Impact Estimation (After Hijack)	Impact Estimation (After Mitigation)	
0	AS16296 TWINIX-AS	AS32645 PIVOT	AS49181 GCX-AS AS1911 DNIC-ASBLK-01910-01914	14.5120%	5.5674%	More Details
1	AS49921 IFOREX-ASN	AS137406 ACI-AS-AP	AS62695 VERSO-NETWORKS AS62634 NGP-TELENETWORKS	26.7212%	8.1971%	More Details

Rows per page: 10 1-2 of 2

Top 1000 Vulnerable Autonomous Systems

AS	% Vulnerability
AS14 COLUMBIA-GW	100.0000%
AS22 DNIC-AS-00022	100.0000%
AS25 UCB	100.0000%
AS26 CORNELL	100.0000%
AS31 CIT	100.0000%
AS34 UDELNET	100.0000%
AS38 UIUC	100.0000%
AS48 DNIC-AS-00048	100.0000%
AS52 UCLA	100.0000%
AS72 SCHLUMBERGER-AS	100.0000%

Rows per page: 10 1-10 of 1000

Copyright © 2021, CSD UoC - INSPIRE Group (FORTH)

Figure 4.1: BPHS Overview.

- **CIDR IPv4, IPv6 prefixes:** BPHS supports real prefixes for the hijacking simulations. The user should submit the corresponding prefix in CIDR (IPv4 or IPv6) format, otherwise an error message is returned. The real prefixes feature is mandatory, because, when we perform realistic RPKI ROV, we check the ASN-prefix pair with the data of the real RPKI repositories of the 5 RIRs (ARIN, APNIC, AFRINIC, LACNIC, and RIPE).

- **Custom simulation attacks:** In this simulation type, the user before the simulation's launch should define his/her preferences for the ASN and CIDR prefix of the victim, hijacker, helper AS(es), and also the number of simulation's repetitions. If one of the submitted ASNs is not included in simulation topology (i.e., the loaded AS-graph of CAIDA) an error message is returned. We mention that the helper ASes (or anycast ASes as they called) collaborate with the victim AS and help it to mitigate the hijack by announcing a mitigation prefix which should be exact or more specific than hijackers to have a positive effect. In real hijacking scenarios, the traffic attracted by the helper ASes, is redirected to legitimate AS, through a tunneling mechanism [29]. In our simulator, we abstract this process, assuming that the traffic attracted by the helper ASes "virtually" belongs to the victim AS.

- **Random simulation attacks:** In this simulation type, contrary to custom simulations, BPHS is responsible to randomly pick the ASN of the victim, hijacker, and helper AS(es) from the AS-graph. Similarly with custom simulations, the user is enabled to launch all the available BGP prefix hijacking attacks. The simulator generates "virtually" prefixes for each attack type; for example, if a user wants to simulate a random sub-prefix hijack then the simulator assumes the prefix "x.y.z.w/24" for the randomly selected victim, and the "x.y.z.w/25" for the randomly selected attacker, helper ASes. In random simulation scenarios, we assume that the victim and helper ASes announce the longest prefix that could partially or fully mitigate the attack. We mention that the random simulation attacks do not support realistic RPKI ROV, due to the "virtually" prefixes that are picked by BPHS. The user is enabled to select the number of random victim-hijacker-helper ASes pairs that BPHS will pick randomly and the simulation repetitions of each selected pair.

- **Real-time RPKI ROV:** In this ROV type, when one node in the generated AS-graph receives a BGP route from a neighboring AS and this node applies RPKI filtering, then it performs realistic route origin validation using the Rootinator tool (see section 2.3.3.1). If the ROV result returned by Rootinator is *"valid"*, then the node accepts the route, else if it is *"invalid"*, then discards it. In the case of an *"unknown"* result, the simulator accepts or discards the route with probability 0.5 . We remind that Routinator periodically connects to the Trust Anchors of the five Regional Internet Registries (RIRs) — APNIC, AFRINIC, ARIN, LACNIC and RIPE NCC — downloads all of the certificates and ROAs in the various repositories, verifies the signatures and makes the valid ROAs available for ROV.

- **Static RPKI ROV:** In this ROV type, contrary to Real-time, the ROV enabled nodes of the generated AS-graph perform RPKI filtering using predefined decisions of the simulator. Specifically:

  - When an ROV enabled node receives a route including the selected victim's prefix and ASN as origin AS in *AS_PATH*, then the ROV result is *"valid"* or *"unknown"* with probability 0.5.
  - If the route includes the mitigation prefix and the victim's or helper's ASN as origin AS in *AS_PATH*, then the ROV result is *"valid"* or *"unknown"* with probability 0.5.
  - In the case that route includes the hijacker's prefix and the ASN as origin AS in *AS_PATH*, then the ROV result is *"invalid"* or *"unknown"* with probability 0.5.

- **Different ROV modes:** We define as ROV mode, the model that BPHS follows to decide which nodes in the simulation AS-graph will perform RPKI ROV. Currently, we support 4 basic ROV modes (and some *"backdoor"* modes for our experiments that are not offered through the web interface) that are described below:
  - ***All:*** all the ASes in the simulation graph perform ROV.
  - ***Random_20:*** BPHS selects randomly 20% of the total number of ASes that will perform ROV.
  - ***ROV deployment monitor:*** BPHS selects the ASes that will perform ROV according to the publicly available dataset of the ROV deployment monitor [50]. The dataset includes the % certainty that an AS performs ROV, according to the results of the real passive measurements that were conducted on the Internet (see methodology approach in Reuter *et al* [116]). First, BPHS selects from the dataset only the ASes that have certainty values greater than 0.5. Then, for each AS in the resulting subset dataset, BPHS checks if it exists in simulation topology; if yes, it is marked as ROV enabled AS.
  - ***Active Measurements:*** BPHS selects the ASes that will perform ROV according to research results of Rodday *et al* [117] which is the latest work that (a) reviews all the previous works that try to identify the ASes that deploy ROV using control plane as well as data plane measurements in the Internet and (b) extends the current state-of-the-art work focusing on controlled data plane measurements to identify the ASes that use RPKI ROV. The publicly available dataset (see [45] 19-07-2021), includes 206 unique ASes performing ROV of which 146 are fully and 60 are partially filtering. For each available AS in the dataset, BPHS checks if it exists in simulation topology; if yes, it is marked as ROV enabled AS.
- **Parallel Simulations:** BPHS uses a very powerful python module, called *mpipe*, enabling the execution of parallel, multi-stage pipeline algorithms. The basic principle of mpipe is the *"worker"*, which can be defined as a separate process that runs an instance of our simulation algorithm. The user, in the tool's setup, can define the number of workers that BPHS will use during the simulation process. The choice depends on the computational power of the server (or desktop) that BPHS will be installed (e.g., more workers on a powerful server). If BPHS has been configured to use 1 worker and the user wants to launch a simulation that requires 4 repetitions, then all repetitions will be executed in serial. In the scenario that BPHS uses 2 workers and the user wants to launch a simulation that requires 4 repetitions, then the load of the first 2 repetitions will be shared between the 2 workers (i.e., each worker will execute 1 repetition). When one of the 2 workers completes its simulation, then asks for the next simulation, if it exists. In the

last scenario, we obtain quicker simulation results, due to parallel execution (assuming the same experimental test-bed).

- **Useful simulation results:** For each successful BGP hijacking simulation, BPHS exports useful results that help the user to easily assess the impact of a specific attack on the Internet (i.e., on each AS or Country). Below, we list useful information that exported from each simulation:
  - ***Infected ASes + paths:*** BPHS, identifies in the generated AS-graph the infected ASes and the infected AS_PATH of each AS (a) before hijacking attempt, (b) after hijacking attempt and (c) after hijacking mitigation by helper and victim ASes. An AS is characterized as infected **before hijack** (useful metric only for exact-prefix hijacks), if its AS_PATH to the victim's prefix includes the AS-number of the hijacker AS. An AS is characterized as infected **after hijack** (useful metric for both exact and sub-prefix hijacks), if its AS_PATH to the hijacker's prefix includes the AS-number of the hijacker AS. An AS is characterized as infected **after mitigation**, if its AS_PATH to the mitigation prefix includes the AS-number of the hijacker AS. (assuming that the mitigation prefix is equal with hijacker's prefix).
  - ***Impact Estimation:*** BPHS, estimates the impact of a hijack on the simulation AS-graph (a) before/after the attack and (b) after the mitigation by helper and victim ASes. To measure the impact of a hijack, BPHS marks a random portion of the ASes in the AS-graph as *"monitor"* ASes and checks how many of them have an infected AS_PATH before/after the attack and after the mitigation, using the information that has been collected, accordingly (see the previous bullet). By default, BPHS mark as *monitor* ASes, the ASes of the graph that have an AS_PATH to the victim's prefix before hijack.
  - ***AS Vulnerability Ranking:*** For each successful simulation, BPHS computes a ranking with the top 1000 vulnerable ASes (after the hijack attempts) for the attacking scenarios that were conducted. BPHS, extracts the ranking by creating an index that has as key the AS-number of an infected AS and as value a floating number that shows the percentage of attacking scenarios in which this AS was identified as infected (e.g., 10/20 of the attacking scenarios). We mention that the AS vulnerability ranking makes sense only if a simulation includes a big sample of hijacking scenarios (e.g., we need a random simulation with more than 20 victim-hijacker pairs).
  - ***Country Vulnerability Ranking:*** For each successful simulation, BPHS computes a country vulnerability ranking (after the hijack attempts) for the attacking scenarios that were conducted. BPHS, extracts the ranking using (a) the data of the AS vulnerability ranking (see the previous bullet) and (b) the AS-info dataset provided by the

CAIDA [5] (shows the country that each AS belongs to). BPHS, creates an index that has as key the vulnerable country and as value a floating number that shows the average vulnerability percentage of the ASes belonging to this country and were identified as infected. Similarly to the AS vulnerability ranking, country vulnerability ranking makes sense only if a simulation includes a big sample of hijacking scenarios.

- **REST API:** In addition to Web Interface, the user can interact with BPHS, using directly the REST API that is provided. This feature is very useful when a different application wants to integrate the functionality of BPHS in its workflow (e.g., the raw results of a requested simulation).

$$\boxed{impact\_estimation = \frac{\#infected\_monitors}{\#total\_monitors}}$$

Table 4.1: Impact estimation of a hijack: computation formula.

$$\boxed{AS\_vulnerability\_score = \frac{\#simulations\_identified\_infected}{\#total\_simulations}}$$

Table 4.2: AS vulnerability score: computation formula.

$$Country\_vulnerability\_score = \frac{1}{n}\sum_{i=1}^{n} s_i = \frac{1}{n}(s_1 + \cdots + s_n)$$

where:

n = #infected ASes belonging to the country,

s_i = the vulnerability score of the i AS

Table 4.3: Country vulnerability score: computation formula.

4.3 Architecture

BPHS is a full-stack web application that inherits all the characteristics of the MVC model (see section 2.6.1). BPHS has been designed in such a way to be easily scalable and user-friendly. Each core component of BPHS (i.e., frontend, backend, database) operates independently from the other components and thus the debugging is easier. BPHS can be installed as a tool on a personal computer or as a service in a publicly available server (the server should have a public IP that is reachable by anyone on the Internet or it could be accessed, through a VPN connection, in a Local Area Network). We have tested BPHS in Ubuntu 18.04 and 20.04 Operating System with no execution problems (or unexpected behaviors).

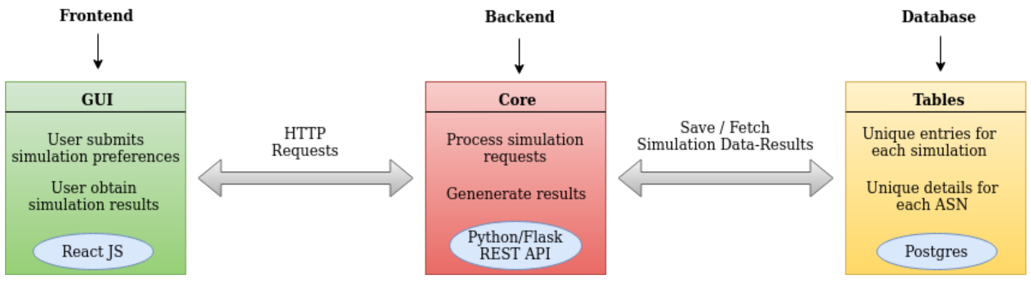

Frontend
Backend
Database
GUI
User submits simulation preferences
User obtain simulation results
React JS
HTTP Requests
Core
Process simulation requests
Genenerate results
Python/Flask REST API
Save / Fetch Simulation Data-Results
Tables
Unique entries for each simulation
Unique details for each ASN
Postgres

Figure 4.2: BPHS Architecture.

In Figure 4.2, we see the architecture of BPHS on an abstract level. The basic components are depicted with a unique color: the *GUI* with green color, the *Core* with red, and the *Tables* with yellow. Next, we describe each component:

- The ***GUI*** component provides a user-friendly interface that is responsible to (a) handle the submitted simulation preferences of the user (e.g., a random simulation with *Active_Measurements* as ROV mode, realistic RPKI filtering and 20 random victim-hijacker repetitions), (b) send (through HTTP POST request) the submitted simulation preferences to the backend (c) request and fetch from the backend (i.e., REST API) the raw simulation results (JSON format), and (d) render properly the raw simulation results using the ReactJS library (see section 2.6.5).

- The ***Core*** component interconnects the frontend with the database component and it has a crucial role in BPHS operation, since it *executes the main logic of the hijacking simulation*. Also, the Core component generates the simulation results and stores them as a unique record (unique key per simulation) to the *bgp_hijacking_simulations* table in the database. The backend provides a REST API that is responsible to (a) properly handle the submitted simulation requests and trigger the simulation scenario, (b) fetch from the database the requested simulation results (in raw format) and send them to the frontend (in JSON format).

- The ***Tables*** component constitutes the database of BPHS that is based on PostgreSQL (see section 2.6.6). To fetch and store information related to launched/completed simulations, BPHS uses a postgres database called *bgp_simulator*. The bgp_simulator database includes 2 basic tables: the *bgp_hijacking_simulations* and *asn_to_org* table. The *bgp_hijacking_simulations* table stores the simulation data/results as a unique record (unique key per simulation). The *asn_to_org* table stores, in tabular format, the data provided by CAIDA's AS to Organization mappings dataset [5]. Each row includes information for a unique ASN (e.g., name, organization name, country, RIR) that is very useful in the rendering process of the simulation results and the experimental evaluation that could be performed.

In the next subsections, we analyze the functionality of the frontend, backend, and database components with more technical details. Also, we describe the hijacking simulation process in detail and we present BPHS GUI.

4.3.1 Backend

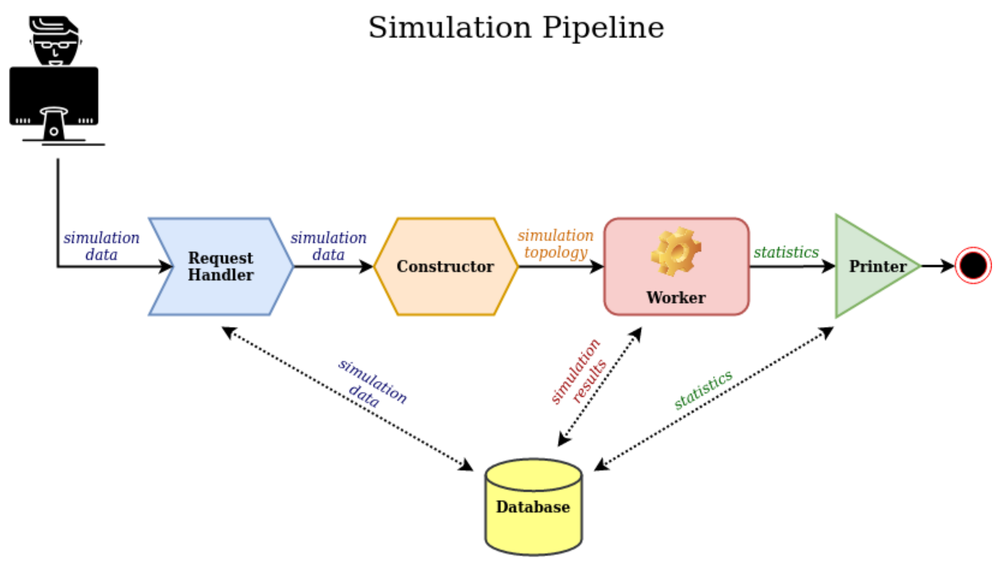


Figure 4.3: The simulation pipeline of BPHS and the exported data of each stage.

In Figure 4.3, we can see the simulation pipeline (or workflow) of the backend component. The pipeline has four stages (i) *Request Handler*, (ii) *Constructor*, (iii) *Worker*, and (iv) *Printer*; each stage is depicted with a unique color. Next, we describe the function of each stage:

- The ***Request Handler***, is responsible for (a) parsing and validating the submitted simulation data, and (b) triggering the simulation. Specifically, BPHS uses the *"RequestParser"* class of the Flask framework to parse the JSON data of the HTTP request. *"RequestParser"* is configured to check the type of the JSON fields and the required simulation fields; also it returns a help message in case of an unexpected field or a type error. If the first validation process is successful, BPHS performs a second data validation. In the second validation process, BPHS checks if the values of the JSON fields satisfy the simulation requirements; for example, BPHS checks if the submitted AS-numbers for the victim, hijacker, and helper ASes exist in the simulation topology or if the submitted IP-prefixes are in CIDR format (required for the real-time RPKI filtering). In case of an invalid value, an HTTP error message (400 error code) is returned. After the successful validation process, BPHS connects to the database to save the simulation data. In response to the successfully created table entry for the simulation data, the database returns to RequestHandler a *unique UUID* for the simulation. The *simulation UUID* has a crucial role in the simulation process because, in the following stages of the pipeline, this id is used for fetching or updating the simulation statistics/results. The next task of RequestHandler is the initialization of a *"Stage"* object of the *mpipe* Python module to define (a) the max number of *workers*, (for enabling the parallel execution of simulation's repetitions), and (b) the Python class of the worker (i.e., the starting point of the worker - in our case the *Constructor* stage). The last task of RequestHandler is to assign each simulation repetition to an available/free worker (i.e., not executing any simulation). If the user submitted a *custom* simulation type, then the same simulation data would be shared across all the workers. If the user submitted a *random* simulation type, then each worker would take different input simulation data, due to the random choice of victim, hijacker, and helper ASes from the simulation topology. Finally, BPHS sends an HTTP message (200 code) to the user, informing of the successful simulation launch.

- The ***Constructor***, is responsible to generate the simulation topology from the CAIDA AS-Relationships and IXPs datasets. First, BPHS parses the AS-Relationships dataset and for each line checks if the specific ASes exists in the topology. If any of them do not exist in the graph, then adds the new node(s); next adds a link between the nodes and marks their relationship. The IXPs dataset adds more links in the AS-graph and thus the simulation results are more realistic. We assume that the ASes being interconnected through an IXP network have a peer-to-peer relationship. Moreover, we ignore the ASes that appear only in the IXP dataset and have a peer-to-peer link through an IXP with an AS that exists in AS-relationships. The second task of the Constructor is to tag the ROV-enabled ASes in the generated topology, according to the submitted user preferences. Each AS in the simulation topology is an instance of the *"BGPnode"* Python class. The object instances of the BGPnode have an instance variable called *"rov"* (boolean), which tags the specific node as ROV enabled (the default state is false, which means that the ASes not supporting RPKI filtering). The third task of the Constructor is the generation of the RPKI ROV table for the submitted simulation scenario and the assignment of that table to the ROV-enabled ASes in the generated topology. *With this approach, the validation time is reduced, because we pre-compute the RPKI ROV table before the hijacking simulation (useful especially on the real-time validation, due to Rootinator latency).* The last task is to initiate the *Worker* giving as input the simulation topology.

- The ***Worker***, is the most important component of BPHS, because it executes the hijacking scenarios and models the BGP protocol. The *Worker* executes the hijacking scenario in three stages: (1) the legitimate announcement by the victim AS, (2) the hijacker announcement, (3) the mitigation by the victim and helper ASes. For each stage, the worker computes and saves, into a dictionary, (a) the number of the infected ASes, (b) a list with the infected ASes, (c) the infected paths of the hijacked ASes and (d) the impact estimation of the hijack. With the end of the final repetition of the simulation scenario, all the collected simulation results are inserted into the database. Next, we describe the propagation process of a BGP announcement, the import/export policies that are used by the ASes and the best path selection algorithm of BPHS:

BGP announcement propagation

Each BGP node in the simulation topology maintains 3 basic data structures, providing information for:

– the neighboring ASes and the AS-relationships with them.
– the AS paths towards all the destination prefixes that was learned by the neighboring ASes (known as Routing-Information-Base or RIB table).
– the best routing paths towards all the destination prefixes (known as Forward-Information-Base table or FIB table).

The FIB table is generated from the RIB table using the best path selection algorithm of the simulator.

When a BGPnode in the simulation topology wants to announce a prefix (BGP route) to its neighbors, first, it searches in the "neighbors" dictionary for the candidate neighbors by filtering them using the export policies algorithm of our simulator. Second, it updates the AS_PATH of the BGP route by adding its AS-number at the start of the AS-path list (e.g., [*ASN*, 3, 2, 1]). Finally, the BGP node announces the BGP route by triggering the inbound filtering mechanism of the candidate neighbor (i.e., to accept or discard the route using the inbound policies).

Import Policies

A BGP node is allowed to receive a BGP route if:

1. the prefix does not belong to the node.
2. the prefix is not hijacked by the node.
3. the AS_PATH does not contain the node's ASN (for loop avoidance).
4. the RPKI ROV result returns *true*.

otherwise, the received BGP route must be discarded.

Best path selection algorithm

A BGP node X selects a path to Y from the set of paths it learns from its neighbors as follows:

1. **Local Preference.** Prefer outgoing paths where the next hop is a customer over outgoing paths where the next hop is a peer over paths where the next hop is a provider (Gao Rexford rules).
2. **Shortest Paths.** Among paths with the highest local preference, prefer shortest paths (i.e., fewest AS hops).
3. **Tie Break.** If there are multiple such paths (i.e., with equal LOCAL_PREF and AS_PATH length), the node selects one at random.

Export Policies

The export policies of an AS are based on the Gao-Rexford rules (see section 2.2.6):

- **Exporting to a provider:** In exchanging routing information with a provider, an AS can export its routes and the routes of its customers, but it can not export routes learned from other providers or peers.
- **Exporting to a customer:** In exchanging routing information with a customer, an AS can export its routes, as well as routes learned from its providers and peers to its customers.
- **Exporting to a peer:** In exchanging routing information with a peer, an AS can export its routes and the routes of its customers, but can not export the routes learned from other providers or peers.

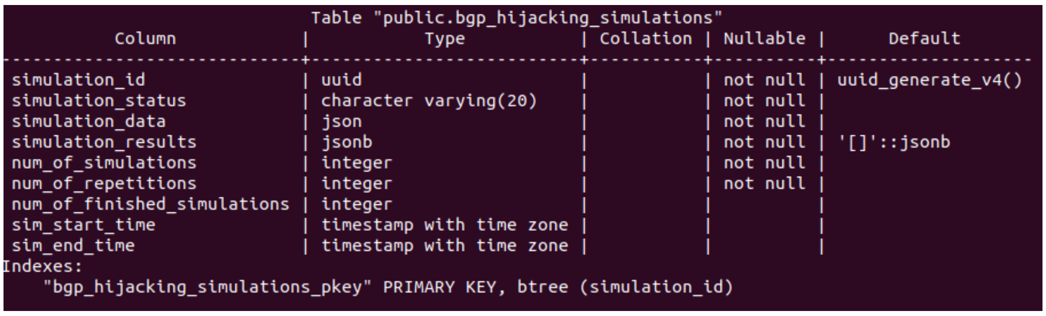
- The ***Printer*** is the last stage of the pipeline and it is responsible to save important statistics for the simulation. The Printer is triggered at the end of the Worker process by the Constructor and checks if the repetition that was executed by the Worker was the last of the total repetitions, using three fields of the bgp_hijacking_simulations table (a) num_of_finished_simulations, (b) nb_of_sims, (c) nb_of_reps. If the num_of_finished_simulations = nb_of_sims * nb_of_reps then the Printer:

  1. updates in the database the simulation status to "Completed" and the end time of the simulation.
  2. exports the simulation results to a JSON file (not required).

4.3.2 Database

The first and most important step of BPHS installation is the generation of a Postgres database, called *BGP_Simulator*, including two tables: (a) *bgp_hijacking_simulations*, and (b) *asn_to_org* table. We have automated the database generation by creating a python script called *"create_db"*. Next we describe the structure of the aforementioned tables:

BGP_HIJACKING_SIMULATIONS Table



```
                           Table "public.bgp_hijacking_simulations"
           Column            |           Type           | Collation | Nullable |      Default
-----------------------------+--------------------------+-----------+----------+--------------------
 simulation_id               | uuid                     |           | not null | uuid_generate_v4()
 simulation_status           | character varying(20)    |           | not null |
 simulation_data             | json                     |           | not null |
 simulation_results          | jsonb                    |           | not null | '[]'::jsonb
 num_of_simulations          | integer                  |           | not null |
 num_of_repetitions          | integer                  |           | not null |
 num_of_finished_simulations | integer                  |           |          |
 sim_start_time              | timestamp with time zone |           |          |
 sim_end_time                | timestamp with time zone |           |          |
Indexes:
    "bgp_hijacking_simulations_pkey" PRIMARY KEY, btree (simulation_id)
```

Figure 4.4: The structure of bgp_hijacking_simulations table.

- ***simulation_id:*** the unique identifier of each submitted simulation; the data type of this field is UUID (or Universal Unique Identifier) [1], which is a 128-bit quantity generated by an algorithm that make it unique in the known universe using the same algorithm (in our case the uuid_generate_v4 function, which generates a UUID value solely based on random numbers).

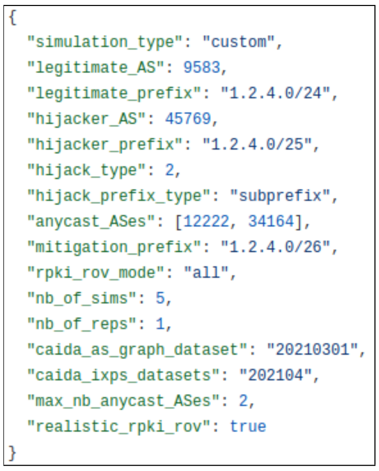
- ***simulation_status:*** indicates the simulation status; there are 3 possible states (a) *In-Progress*, (b) *Completed*, (c) *Failed*; the data type of this field is VARCHAR with max 20 characters.

- ***simulation_data:*** indicates the submitted simulation data; the data type of this field is JSON and in Figure 4.5 we show an example of the JSON's structure.

- ***simulation_results:*** indicates the simulation results; the data type of this field is a JSON array and each JSON object in the array indicates the simulation result of one repetition (see Figure 4.6).

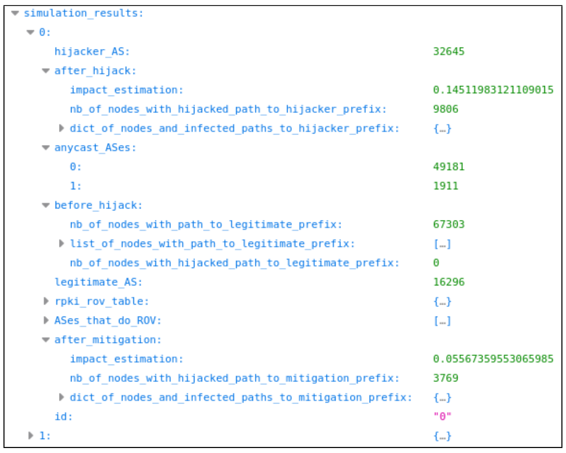
- ***num_of_simulations:*** for a custom simulation type, indicates the number of simulation repetitions of the submitted hijacking scenario; for a random simulation type, indicates the total random hijacking scenarios (i.e., the different pairs of victim-hijacker); the data type of this field is Integer.

- ***num_of_repetitions:*** for a random simulation type, indicates the total repetitions of each random hijacking scenario; for a custom simulation the value should be equal to 1; the data type of this field is Integer.
- ***num_of_finished_simulations:*** indicates the count of the completed simulations/repetitions; the data type of this field is Integer.
- ***sim_start_time:*** indicates the date-time that the simulation data inserted in the table; the data type of this field is "timestamptz" (i.e., timestamp with the time zone).
- ***sim_end_time:*** indicates the date-time that the simulation completed successfully or failed; the data type of this field is "timestamptz" (i.e., timestamp with the time zone).



```
{
  "simulation_type": "custom",
  "legitimate_AS": 9583,
  "legitimate_prefix": "1.2.4.0/24",
  "hijacker_AS": 45769,
  "hijacker_prefix": "1.2.4.0/25",
  "hijack_type": 2,
  "hijack_prefix_type": "subprefix",
  "anycast_ASes": [12222, 34164],
  "mitigation_prefix": "1.2.4.0/26",
  "rpki_rov_mode": "all",
  "nb_of_sims": 5,
  "nb_of_reps": 1,
  "caida_as_graph_dataset": "20210301",
  "caida_ixps_datasets": "202104",
  "max_nb_anycast_ASes": 2,
  "realistic_rpki_rov": true
}
```

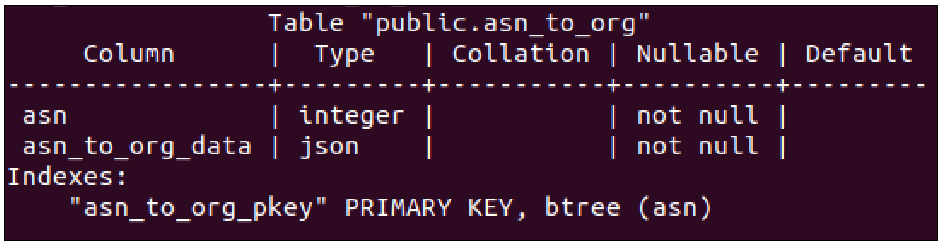
Figure 4.5: Simulation data, JSON structure example.



```
▾ simulation_results:
  ▾ 0:
      hijacker_AS:                                         32645
    ▾ after_hijack:
        impact_estimation:                                 0.14511983121109015
        nb_of_nodes_with_hijacked_path_to_hijacker_prefix: 9806
      ▸ dict_of_nodes_and_infected_paths_to_hijacker_prefix: {…}
    ▾ anycast_ASes:
        0:                                                 49181
        1:                                                 1911
    ▾ before_hijack:
        nb_of_nodes_with_path_to_legitimate_prefix:        67303
      ▸ list_of_nodes_with_path_to_legitimate_prefix:      […]
        nb_of_nodes_with_hijacked_path_to_legitimate_prefix: 0
      legitimate_AS:                                       16296
    ▸ rpki_rov_table:                                      {…}
    ▸ ASes_that_do_ROV:                                    […]
    ▾ after_mitigation:
        impact_estimation:                                 0.05567359553065985
        nb_of_nodes_with_hijacked_path_to_mitigation_prefix: 3769
      ▸ dict_of_nodes_and_infected_paths_to_mitigation_prefix: {…}
      id:                                                  "0"
  ▸ 1:                                                     {…}
```

Figure 4.6: Simulation results, JSON structure example.

ASN_TO_ORG Table

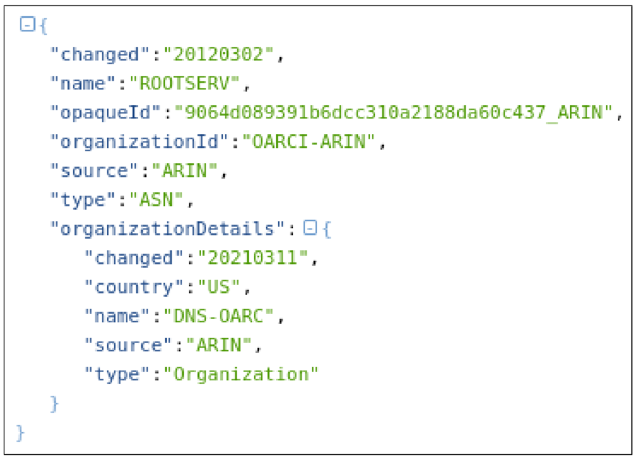


```
                    Table "public.asn_to_org"
     Column      |  Type   | Collation | Nullable | Default
-----------------+---------+-----------+----------+---------
 asn             | integer |           | not null |
 asn_to_org_data | json    |           | not null |
Indexes:
    "asn_to_org_pkey" PRIMARY KEY, btree (asn)
```

Figure 4.7: The structure of asn_to_org table.

- ***asn:*** is the primary key of the table and indicates a unique AS number provided by CAIDA's AS to Organization mappings dataset [5]; the data type of this field is Integer.

- ***as_to_org_data:*** a JSON object including useful information for the ASN (name, RIR) and the Organization (id, name, country, RIR) that the ASN belongs to (provided by CAIDA's AS to Organization mappings dataset [5]). Figure 4.8 depicts an example of the query result on the as_to_org_data field for the ASN 112.



```
⊟{
   "changed":"20120302",
   "name":"ROOTSERV",
   "opaqueId":"9064d089391b6dcc310a2188da60c437_ARIN",
   "organizationId":"OARCI-ARIN",
   "source":"ARIN",
   "type":"ASN",
   "organizationDetails":⊟{
      "changed":"20210311",
      "country":"US",
      "name":"DNS-OARC",
      "source":"ARIN",
      "type":"Organization"
   }
}
```

Figure 4.8: ASN_to_Org_data field, JSON structure example.

4.3.3 Frontend

The Graphical User Interface of BPHS have been developed with ReactJS as a *Single Page* Web *Application (SPA)* [81] that operates independently from the backend component. The basic idea of a SPA is that we can have a full operative application, in just one page, with only one call to the server, without changing the URL, or at least without consuming a new Page. In our React App (as in most of the React Apps), we have an index.js file, which is the starting point of the App. The index.js is responsible to load all the components of our App in the ReactDOM. The core component of our App is called *Router*. The *Router* component decides which component should be rendered; in other words, it is the navigator of our website that helps us to view the different web pages.

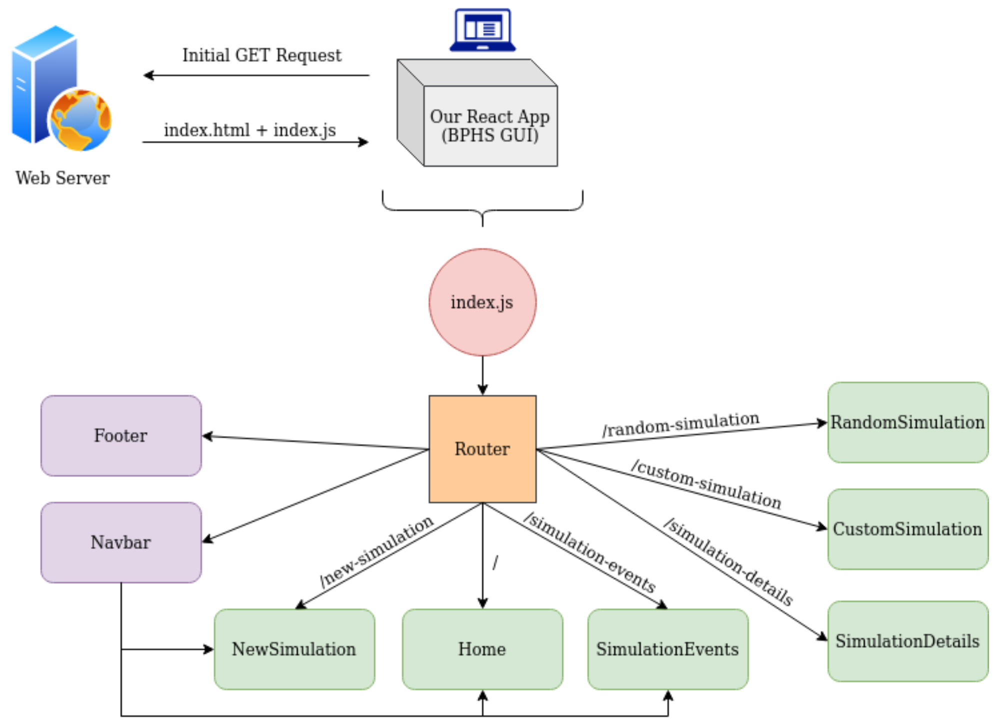


Figure 4.9: Web App (BPHS GUI) Architecture.

In Figure 4.9, we show the architecture of the Web App. The client sends an HTTP GET request to the Web server to fetch the appropriate app files. The Web server sends as reply only 2 files, the index.html, and the index.js which includes all the app's functionality. The client is not required to communicate again with the server (except in the case of browser refresh). The Router component contains eight sub-components that can be rendered. We depict with purple color the components that are rendered statically in GUI and with green color the components/pages that are rendered dynamically in GUI according to user preferences. Next, we present each component/page showing screenshots from the BPHS GUI.

Home, Navbar, Footer

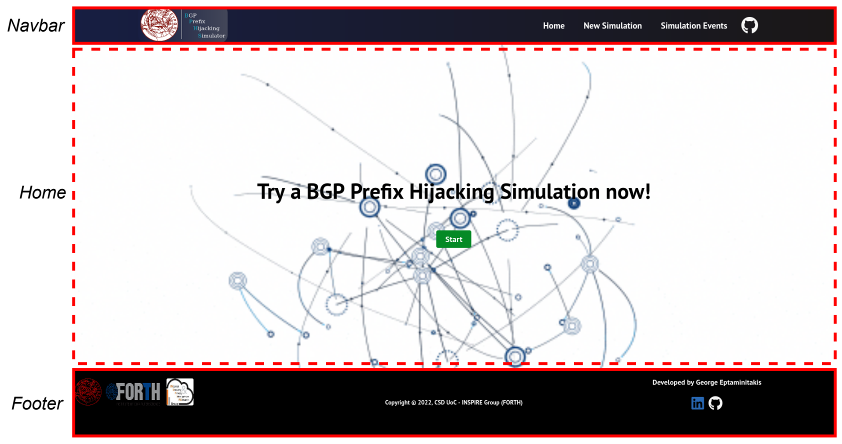

Navbar
Home
Footer
BGP Prefix Hijacking Simulator
Home
New Simulation
Simulation Events
Try a BGP Prefix Hijacking Simulation now!
Start
FORTH
Copyright © 2022, CSD UoC - INSPIRE Group (FORTH)
Developed by George Eptaminitakis

Figure 4.10: The *Home* page of BPHS.

In Figure 4.10, we see the *Home* page of BPHS, the *Navbar* and the *Footer* component. The *Navbar* contains hyperlinks that redirect the user to *Home*, *New Simulation* and *Simulation Events* page. Also it contains a link to BPHS github repository. The *Home* page includes a simple button that redirects the user to the *New Simulation* page. The *Footer* includes clickable images redirecting the user to external pages (e.g., CSD department UoC, Linked In profile of BPHS developer).

New Simulation

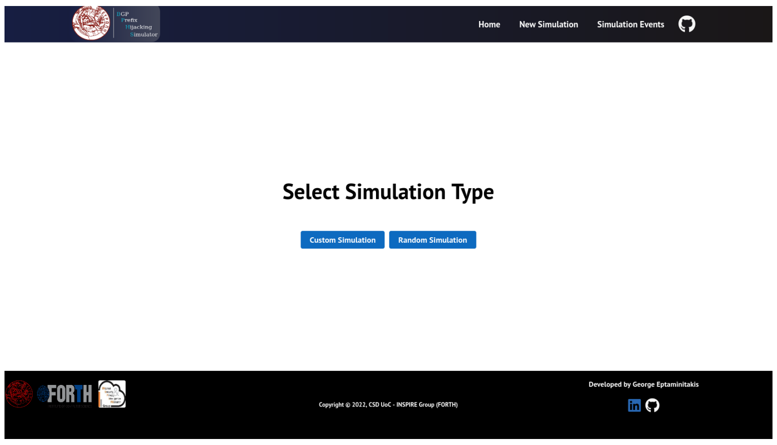
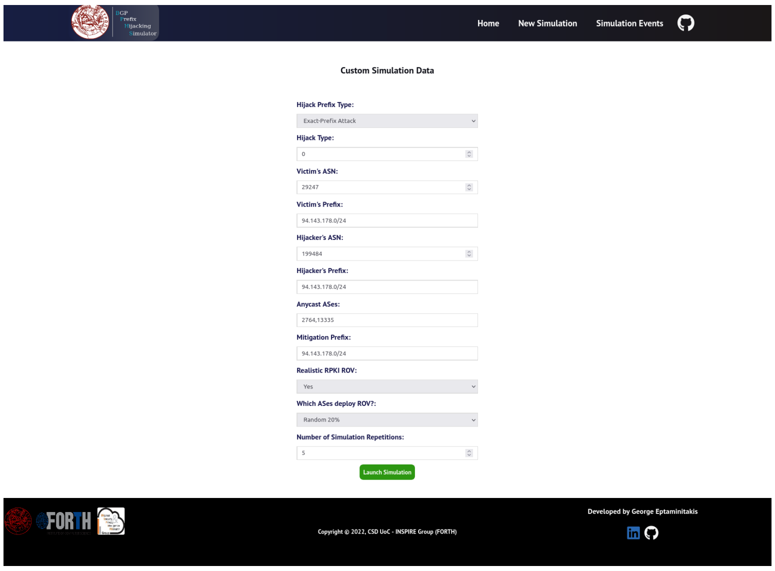

BGP Prefix Hijacking Simulator
Home
New Simulation
Simulation Events
Select Simulation Type
Custom Simulation
Random Simulation
FORTH
Copyright © 2022, CSD UoC - INSPIRE Group (FORTH)
Developed by George Eptaminitakis

Figure 4.11: The *New Simulation* page of BPHS.

In Figure 4.11, we see the *New Simulations* page, which contains 2 buttons that redirect the user to *Custom Simulation* and *Random Simulation* page respectively.

Custom Simulation, Random Simulation

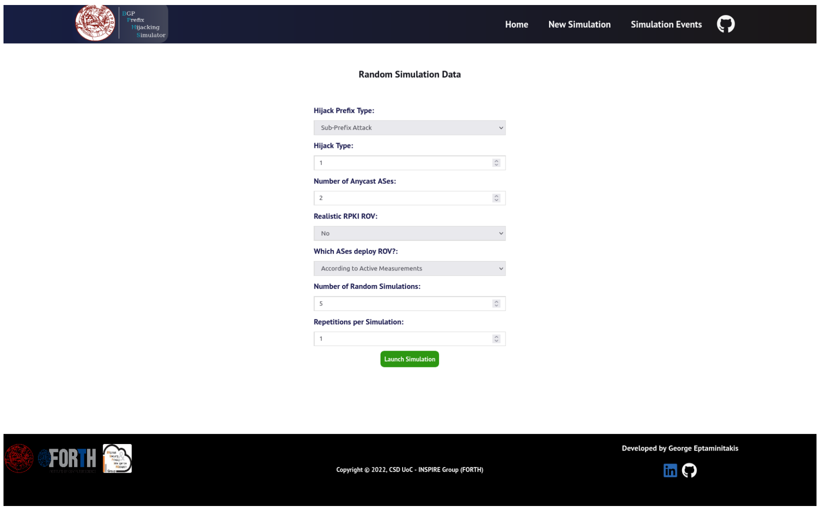


Custom Simulation Data

Hijack Prefix Type: Exact-Prefix Attack
Hijack Type: 0
Victim's ASN: 29247
Victim's Prefix: 94.143.178.0/24
Hijacker's ASN: 199484
Hijacker's Prefix: 94.143.178.0/24
Anycast ASes: 2764,13335
Mitigation Prefix: 94.143.178.0/24
Realistic RPKI ROV: Yes
Which ASes deploy ROV?: Random 20%
Number of Simulation Repetitions: 5

Copyright © 2022, CSD UoC - INSPIRE Group (FORTH)
Developed by George Eptaminitakis

Figure 4.12: The *Custom Simulation* page of BPHS.



Random Simulation Data

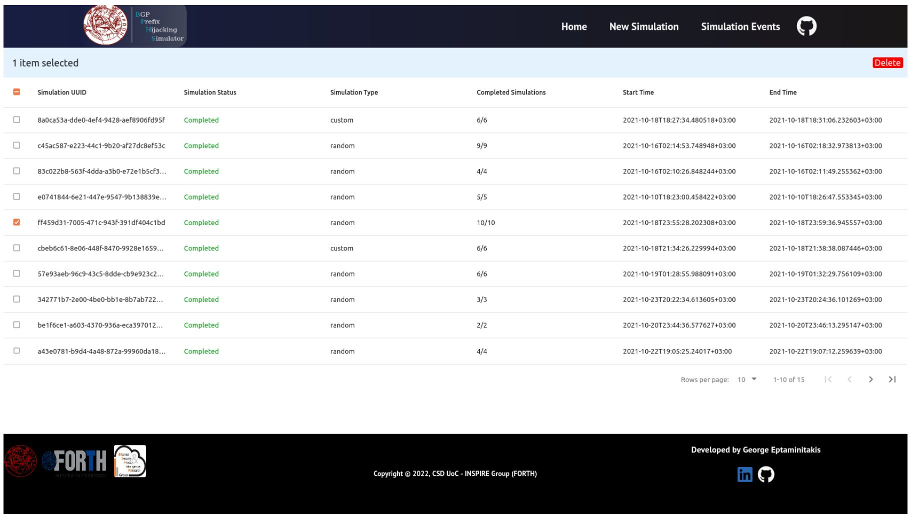
Hijack Prefix Type: Sub-Prefix Attack
Hijack Type: 1
Number of Anycast ASes: 2
Realistic RPKI ROV: No
Which ASes deploy ROV?: According to Active Measurements
Number of Random Simulations: 5
Repetitions per Simulation: 1

Copyright © 2022, CSD UoC - INSPIRE Group (FORTH)
Developed by George Eptaminitakis

Figure 4.13: The *Random Simulation* page of BPHS.

In Figure 4.12 and 4.13, we see the *Custom/Random Simulation* pages, which contain a form that helps users make their preferences and submit the simulation data (see section 4.2 for understanding the form's fields) using the *Launch Simulation* button. In case of an unexpected field value or network error (e.g., unable to connect with the backend server), a help or error message is displayed. In case of successful form submission, the user redirects to *Simulation Events* page.

Simulation Events, Simulation Details



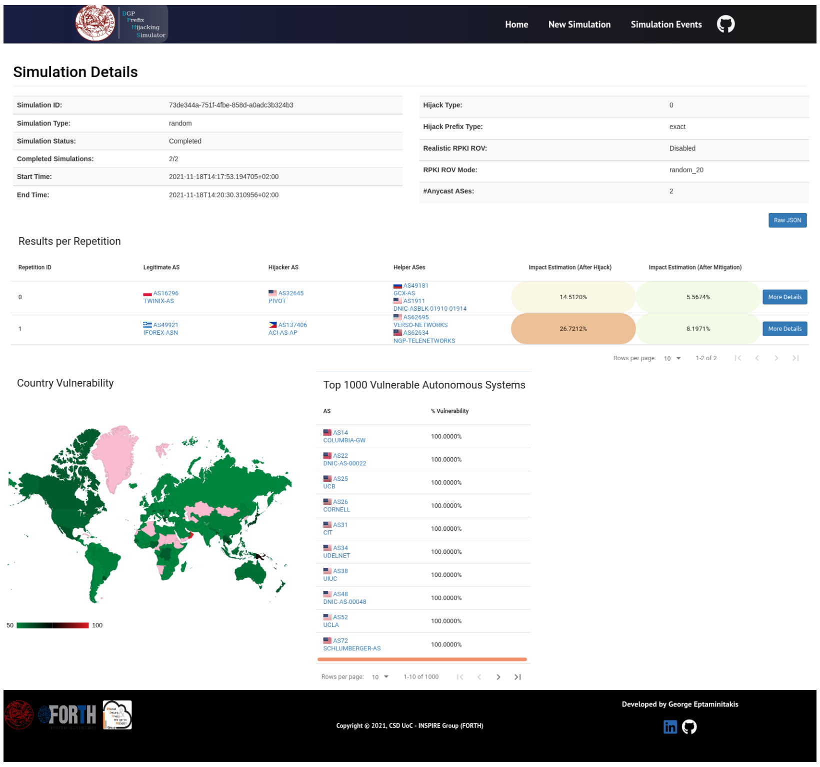
1 item selected

	Simulation UUID	Simulation Status	Simulation Type	Completed Simulations	Start Time	End Time
☐	8a0ca53a-dde0-4ef4-9428-aef8906fd95f	Completed	custom	6/6	2021-10-18T18:27:34.480518+03:00	2021-10-18T18:31:06.232603+03:00
☐	c45ac587-e223-44c1-9b20-af27dc8ef53c	Completed	random	9/9	2021-10-16T02:14:53.748948+03:00	2021-10-16T02:18:32.973813+03:00
☐	83c022b8-563f-4dda-a3b0-e72e1b5cf3...	Completed	random	4/4	2021-10-16T02:10:26.848244+03:00	2021-10-16T02:11:49.255362+03:00
☐	e0741844-6e21-447e-9547-9b138839e...	Completed	random	5/5	2021-10-10T18:23:00.458422+03:00	2021-10-10T18:26:47.553345+03:00
✓	ff459d31-7005-471c-943f-391df404c1bd	Completed	random	10/10	2021-10-18T23:55:28.202308+03:00	2021-10-18T23:59:36.945557+03:00
☐	cbeb6c61-8e06-448f-8470-9928e1659...	Completed	custom	6/6	2021-10-18T21:34:26.229994+03:00	2021-10-18T21:38:38.087446+03:00
☐	57e93aeb-96c9-43c5-8dde-cb9e923c2...	Completed	random	6/6	2021-10-19T01:28:55.988091+03:00	2021-10-19T01:32:29.756109+03:00
☐	342771b7-2e00-4be0-bb1e-8b7ab722...	Completed	random	3/3	2021-10-23T20:22:34.613605+03:00	2021-10-23T20:24:36.101269+03:00
☐	be1f6ce1-a603-4370-936a-eca397012...	Completed	random	2/2	2021-10-20T23:44:36.577627+03:00	2021-10-20T23:46:13.295147+03:00
☐	a43e0781-b9d4-4a48-872a-99960da18...	Completed	random	4/4	2021-10-22T19:05:25.24017+03:00	2021-10-22T19:07:12.259639+03:00

Rows per page: 10 1-10 of 15

Copyright © 2022, CSD UoC - INSPIRE Group (FORTH)

Figure 4.14: The *Simulation Events* page of BPHS.



Simulation Details

Simulation ID:	73de344a-751f-4fbe-858d-a0adc3b324b3
Simulation Type:	random
Simulation Status:	Completed
Completed Simulations:	2/2
Start Time:	2021-11-18T14:17:53.194705+02:00
End Time:	2021-11-18T14:20:30.310956+02:00

Hijack Type:	0
Hijack Prefix Type:	exact
Realistic RPKI ROV:	Disabled
RPKI ROV Mode:	random_20
#Anycast ASes:	2

Results per Repetition

Repetition ID	Legitimate AS	Hijacker AS	Helper ASes	Impact Estimation (After Hijack)	Impact Estimation (After Mitigation)	
0	AS16296 TWINIX-AS	AS32645 PIVOT	AS49181 GCX-AS AS1911 DNIC-ASBLK-01910-01914	14.5120%	5.5674%	More Details
1	AS49921 IFOREX-ASN	AS137406 ACI-AS-AP	AS62695 VERSO-NETWORKS AS62634 NGP-TELENETWORKS	26.7212%	8.1971%	More Details

Rows per page: 10 1-2 of 2

Country Vulnerability

Top 1000 Vulnerable Autonomous Systems

AS	% Vulnerability
AS14 COLUMBIA-GW	100.0000%
AS22 DNIC-AS-00022	100.0000%
AS25 UCB	100.0000%
AS26 CORNELL	100.0000%
AS31 CIT	100.0000%
AS34 UDELNET	100.0000%
AS38 UIUC	100.0000%
AS48 DNIC-AS-00048	100.0000%
AS52 UCLA	100.0000%
AS72 SCHLUMBERGER-AS	100.0000%

Rows per page: 10 1-10 of 1000

Copyright © 2021, CSD UoC - INSPIRE Group (FORTH)

Figure 4.15: The *Simulation Details* page of BPHS.

In Figure 4.14, we see the *Simulation Events* page, which contains a table with brief information for each submitted simulation. The table has 6 columns showing information for the UUID (i.e, unique id) of the simulation, the simulation status, the simulation type, the number of completed simulations and the start/end time of the simulation. The user is enabled to click on each row for exploring the simulation results (redirection to *Simulation Details* page) or to delete the simulation/row by clicking the checkbox at the left corner and then the *Delete* button that appears in the top corner of the table. The simulation events can be sorted according to a specific column (e.g., start time) by clicking the specific column name.

In Figure 4.15, we see the structure of the *Simulation Details* page. This page displays all the simulation results starting from the top corner with a table including the simulation statistics/data. We mention that by clicking on the victim, hijacker, or mitigation prefix, BPHS redirects us to the RIPEstat service showing more information for the clicked prefix. Under the statistics table, we observe a button called *Raw JSON*. This button enables us to get all the simulation results/data in JSON format (in a new browser tab). The *Results per Repetition* enable us to get the simulation results of each repetition. Specifically, each row shows the selected legitimate, hijacker, helper ASes and the impact estimation value after the hijacking attempt and after the mitigation attempt. We mention that the Legitimate AS, Hijacker AS, Helper AS columns includes hoverable hyperlinks as values, enabling (a) to get more information for the hovered AS (e.g., country, name, organization name/id, RIR) and (b) redirection to AS-Rank service showing more information for the clicked AS. By clicking the *More Details* button a pop-up window is displayed in the foreground including the repetition results in detail.

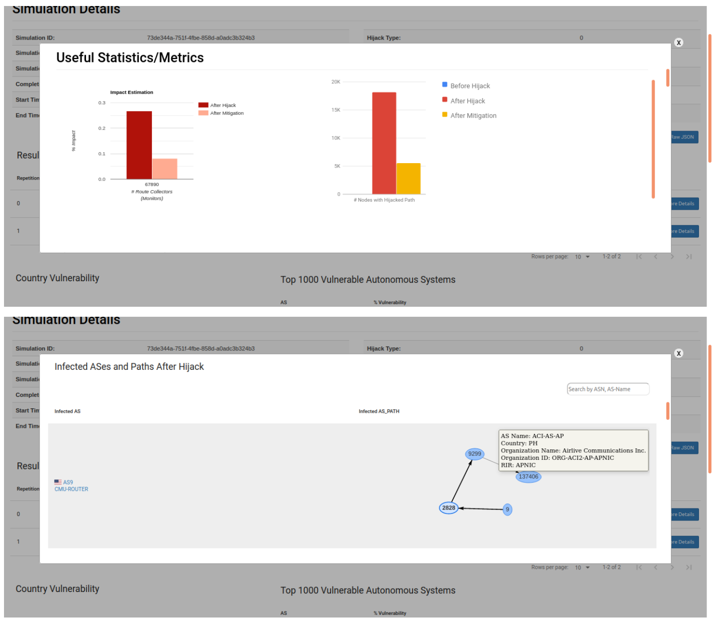


Figure 4.16: Detailed results per simulation (pop-up sample).

In Figure 4.16, we see a sample of the pop-up information. The pop-up provides (a) the RPKI ROV table used by the ASes during the simulation, (b) a table including the ASes applying RPKI filtering, (c) bar-plots showing useful statistics/metrics, (d) two tables including the infected ASes and the infected AS_PATH of each AS (displayed as interactive AS-graph) after hijack and after mitigation respectively. We mention that all the tables provide a searching filter for easier navigation.

Finally, the *Simulation Details* page displays in a Geo-Chart the Country Vulnerability Ranking using information from all the repetition results and also in a separate table the top 1000 vulnerable ASes in descent order using information from all the repetition results (see section 4.2 for more technical details on the ranking approach).

4.4 Challenges

Next, we list the main challenges that we faced during BPHS development:

- **Realtime ROV.** In the first backend version of BPHS, if the selected RPKI ROV type was "Realistic", each ROV enabled AS in the simulation topology was performing an HTTP GET Request to Rootinator to validate each incoming BGP route. This operation was extremely time-consuming, due to the multiple HTTP requests that were handled concurrently by Rootinator resulting in late replies (most requests were aborted by Rootinator). We tackled this problem by pre-computing and assigning the RPKI ROV table to the ROV-enabled ASes before the hijacking simulation. In the old version, the same ROV operations were repeated from all BGPnodes, however, in the updated version all the possible ROV operations are pre-computed only once.

- **Hijacker has no path to Victim.** In BPHS, by default, the Hijacker AS is required to have an AS_PATH to Victim AS before the attack, because in Type {1,..,N} attacks the malicious AS_PATH is constructed from the Victim-Hijacker AS_PATH. In case the Hijacker has no path to Victim AS the routing information on each BGP node is cleared and the simulation/repetition is repeated again (BPHS terminates the simulation/repetition when detects an infinitive loop).

- **Mpipe memory leak.** Each mpipe worker (Linux process) allocates the needed memory space for simulation execution. At the end of the simulation the worker should release the allocated memory. The problem with our first implementation was that the worker was not releasing the allocated memory at the end of the simulation but in Flask server termination. We tackled this problem by adding a special termination signal provided by mpipe library (i.e., pipe.put(None)).

- **Cross-Origin Request Blocked.** Cross-Origin Resource Sharing (CORS) is a standard that allows a server to relax the same-origin policy [21]. This is used to explicitly allow some cross-origin requests while rejecting others. For example a Flask server could be configured to accept requests only from a specific source IP address and port. To avoid CORS problems, we have configured (a) the npm server [41] of BPHS GUI to redirect to Flask server (port 5000) any request it receives on its port 3000 and destined to a different domain (i.e., IP and port) and, (b) the backend Flask app to use the Flask-CORS module extension making cross-origin AJAX possible.

Chapter 5

Evaluation

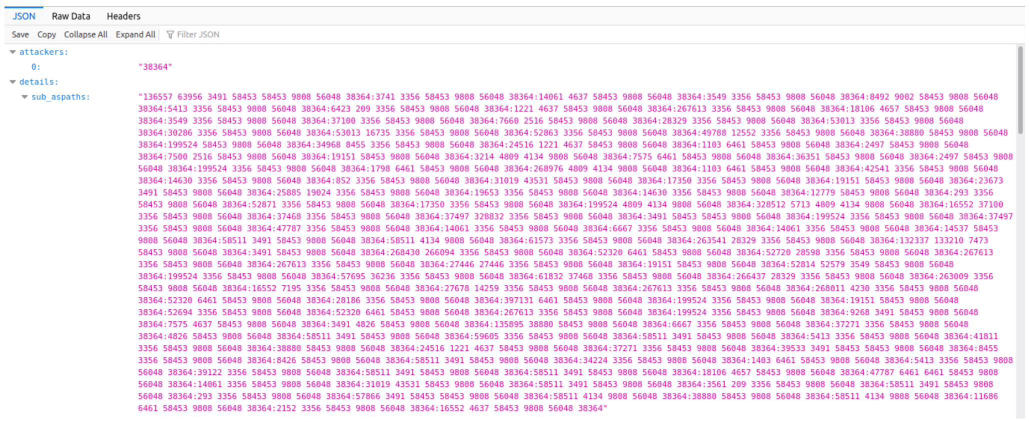
In this chapter, we evaluate BPHS applying a number of experimental scenarios. First, we replay real historical hijacks that were detected on the Internet using data from the hijacking observatory of CAIDA (see section 2.4.3) and we check the BPHS ability to detect the same hijacked ASes as detected in CAIDA datasets. Then, we show through a comparative study the RPKI adoption benefits on the Internet's backbone and we present useful insights extracted during the simulation experiments (e.g., AS - Country vulnerability ranking list from a large set of successful simulations). Finally, we measure the execution time and the memory requirements of each simulation according to mpipe *Workers*.

5.1 Replaying Real Hijacks

The first question that we tried to answer is the following: *Are the simulation results realistic? (i.e., closer to the real Internet)*. The only way to measure the *"realistic"* level of BPHS is to replay real Internet hijacking scenarios in the simulator. Next, we present the methodology that we followed to answer the above question and the experimental results.

5.1.1 Methodology

The only publicly available service ideal for the purposes of this experiment is the CAIDA hijacks observatory, because it provides real data-plane information for thousands of real hijacking attempts on the Internet that can be compared with the BPHS simulation results. Specifically, for each suspicious event, the Observatory augments control-plane (BGP) data with data-plane measurements (traceroutes) executed from a set of RIPE Atlas probes [46], on-the-fly and save the results in JSON format (i.e., the AS_PATH (benign or malicious) of each RIPE Atlas probe (monitor AS) towards victim prefix after the attack). The JSON object includes all the information to replay the hijack (i.e., victim, hijacker {ASN, prefix}). The only problem that we faced during the experiment setup was the collection of the hijacking data, due to the lack of an API to automatically fetch hijacking events from CAIDA observatory. We tackled this problem by manually fetching 64 suspicious Type-0 events (detected in October 2021) using *curl* command. Next, we list the experiment steps.



```
▾ attackers:
      0:                 "38364"
▾ details:
   ▾ sub_aspaths:        "136557 63956 3491 58453 58453 9808 56048 38364:3741 3356 58453 9808 56048 38364:14061 4637 58453 9808 56048 38364:3549 3356 58453 9808 56048 38364:8492 9002 58453 9808 56048
                         38364:5413 3356 58453 9808 56048 38364:6423 209 3356 58453 9808 56048 38364:1221 4637 58453 9808 56048 38364:267613 3356 58453 9808 56048 38364:18106 4657 58453 9808 56048
                         38364:3549 3356 58453 9808 56048 38364:37100 3356 58453 9808 56048 38364:7660 2516 58453 9808 56048 38364:28329 3356 58453 9808 56048 38364:53013 3356 58453 9808 56048
                         38364:30286 3356 58453 9808 56048 38364:53013 16735 3356 58453 9808 56048 38364:52863 3356 58453 9808 56048 38364:49788 12552 3356 58453 9808 56048 38364:38880 58453 9808 56048
                         38364:199524 58453 9808 56048 38364:34968 8455 3356 58453 9808 56048 38364:24516 1221 4637 58453 9808 56048 38364:1103 6461 58453 9808 56048 38364:2497 58453 9808 56048
                         38364:7500 2516 58453 9808 56048 38364:19151 58453 9808 56048 38364:3214 4809 4134 9808 56048 38364:7575 6461 58453 9808 56048 38364:36351 58453 9808 56048 38364:2497 58453 9808
                         56048 38364:199524 3356 58453 9808 56048 38364:1798 6461 58453 9808 56048 38364:268976 4809 4134 9808 56048 38364:1103 6461 58453 9808 56048 38364:42541 3356 58453 9808 56048
                         38364:14630 3356 58453 9808 56048 38364:852 3356 58453 9808 56048 38364:31019 43531 58453 9808 56048 38364:17350 3356 58453 9808 56048 38364:19151 58453 9808 56048 38364:23673
                         3491 58453 9808 56048 38364:25885 19024 3356 58453 9808 56048 38364:19653 3356 58453 9808 56048 38364:14630 3356 58453 9808 56048 38364:12779 58453 9808 56048 38364:293 3356
                         58453 9808 56048 38364:52871 3356 58453 9808 56048 38364:17350 3356 58453 9808 56048 38364:199524 4809 4134 9808 56048 38364:328512 5713 4809 4134 9808 56048 38364:16552 37100
                         3356 58453 9808 56048 38364:37468 3356 58453 9808 56048 38364:37497 328832 3356 58453 9808 56048 38364:3491 58453 58453 9808 56048 38364:199524 3356 58453 9808 56048 38364:37497
                         3356 58453 9808 56048 38364:47787 3356 58453 9808 56048 38364:14061 3356 58453 9808 56048 38364:6667 3356 58453 9808 56048 38364:14061 3356 58453 9808 56048 38364:14537 58453
                         9808 56048 38364:58511 3491 58453 9808 56048 38364:58511 4134 9808 56048 38364:61573 3356 58453 9808 56048 38364:263541 28329 3356 58453 9808 56048 38364:132337 133210 7473
                         58453 9808 56048 38364:3491 58453 9808 56048 38364:268430 266094 3356 58453 9808 56048 38364:52320 6461 58453 9808 56048 38364:52720 28598 3356 58453 9808 56048 38364:267613
                         3356 58453 9808 56048 38364:267613 3356 58453 9808 56048 38364:27446 27446 3356 58453 9808 56048 38364:19151 58453 9808 56048 38364:52814 52579 3549 58453 9808 56048
                         38364:199524 3356 58453 9808 56048 38364:57695 36236 3356 58453 9808 56048 38364:61832 37468 3356 58453 9808 56048 38364:266437 28329 3356 58453 9808 56048 38364:263009 3356
                         58453 9808 56048 38364:16552 7195 3356 58453 9808 56048 38364:27678 14259 3356 58453 9808 56048 38364:267613 3356 58453 9808 56048 38364:268011 4230 3356 58453 9808 56048
                         38364:52320 6461 58453 9808 56048 38364:28186 3356 58453 9808 56048 38364:397131 6461 58453 9808 56048 38364:199524 3356 58453 9808 56048 38364:19151 58453 9808 56048
                         38364:52694 3356 58453 9808 56048 38364:52320 6461 58453 9808 56048 38364:267613 3356 58453 9808 56048 38364:199524 3356 58453 9808 56048 38364:9268 3491 58453 9808 56048
                         38364:7575 4637 58453 9808 56048 38364:3491 4826 58453 9808 56048 38364:135895 38880 58453 9808 56048 38364:6667 3356 58453 9808 56048 38364:37271 3356 58453 9808 56048
                         38364:4826 58453 9808 56048 38364:58511 3491 58453 9808 56048 38364:59605 3356 58453 9808 56048 38364:58511 3491 58453 9808 56048 38364:5413 3356 58453 9808 56048 38364:41811
                         3356 58453 9808 56048 38364:38880 58453 9808 56048 38364:24516 1221 4637 58453 9808 56048 38364:37271 3356 58453 9808 56048 38364:39533 3491 58453 58453 9808 56048 38364:8455
                         3356 58453 9808 56048 38364:8426 58453 9808 56048 38364:58511 3491 58453 9808 56048 38364:34224 3356 58453 9808 56048 38364:1403 6461 58453 9808 56048 38364:5413 3356 58453 9808
                         56048 38364:39122 3356 58453 9808 56048 38364:58511 3491 58453 9808 56048 38364:58511 3491 58453 9808 56048 38364:18106 4657 58453 9808 56048 38364:47787 6461 6461 58453 9808
                         56048 38364:14061 3356 58453 9808 56048 38364:31019 43531 58453 9808 56048 38364:58511 3491 58453 9808 56048 38364:3561 209 3356 58453 9808 56048 38364:58511 3491 58453 9808
                         56048 38364:293 3356 58453 9808 56048 38364:57866 3491 58453 58453 9808 56048 38364:58511 4134 9808 56048 38364:38880 58453 9808 56048 38364:58511 4134 9808 56048 38364:11686
                         6461 58453 9808 56048 38364:2152 3356 58453 9808 56048 38364:16552 4637 58453 9808 56048 38364"
```

Figure 5.1: Historical hijacking data sample from CAIDA Observatory: the AS_PATH (benign or malicious) of each RIPE Atlas probe towards victim prefix after the attack.

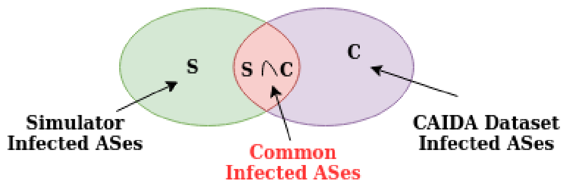
For each historical hijack/dataset:

1. **We replayed the attack in BPHS.** The CAIDA datasets used for the simulations were (a) the October 2021 AS-graph dataset, and (b) the July 2021 IXPs dataset. Furthermore, the RPKI ROV mode that was used was the *"rov_active_measurements"* and the RPKI ROV type was *"realtime"* (see section 4.2 for more information). We mention that some datasets include more than 1 hijacker AS; in that case, we replayed the attack for all victim-hijacker combinations.

2. **We extracted the infected ASes detected by BPHS.** For this task, we queried the simulation results from the database as a JSON object, and we used the field providing a list with the hijacked ASes after the attack.

3. **We identified the infected ASes from the historical dataset.** For this task, we parsed the "aspaths" field (or the "sub_aspaths field for the subprefix attacks") of the historical data. In Figure 5.1, we see the format of the aforementioned fields. It is a string including the AS_PATHs (benign or malicious) of all RIPE Atlas probes towards the victim prefix after the attack, that are separated with the ":" character. A monitor AS (RIPE atlas probe) was tagged as infected, if the last AS in AS_PATH was equal to hijacker AS.

4. **We found the common infected ASes of the above 2 sets.** In this task, we measured the ability of BPHS to identify the infected ASes extracted from the historical dataset. For that purpose, we introduced a metric called *Level of Agreement (LoA)*, defined as the number of common infected ASes from the two sets divided by the number of the infected ASes from the historical dataset (see Figure 5.2).



$$\text{Level of Agreement} = \frac{|S \cap C|}{|C|}$$

Figure 5.2: Replaying real hijacks: Level of Agreement (LoA).

5.1.2 Results

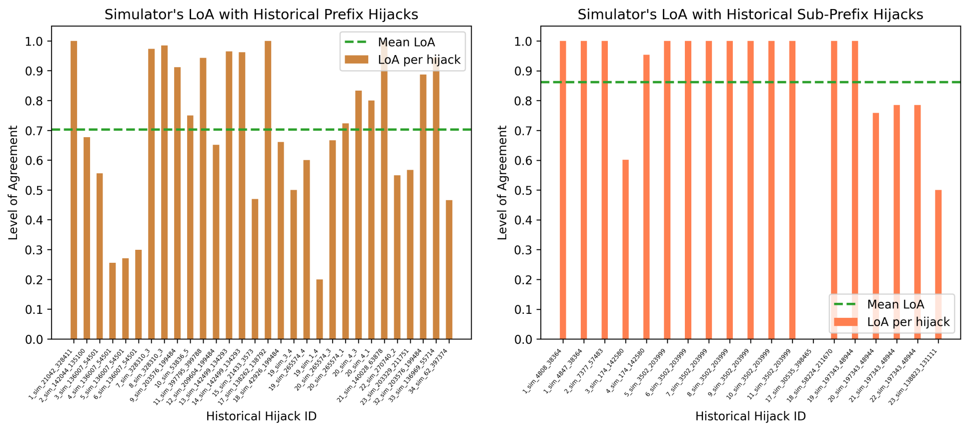


Figure 5.3: Replaying real hijacks: LoA results.

In Figure 5.3, we see the Level of Agreement of BPHS with each replayed historical hijacking attempt and the average LoA score for both prefix and sub-prefix hijacks. The mean LoA score of BPHS for the prefix hijacks is around 70% and 87% for the sub-prefix hijacks. As we expected, the mean LoA score for the sub-prefix events is higher than the prefix events, because in sub-prefix hijacks the attacker's success rate (i.e, the impact) is higher and the probability to hit a common infected AS increasing (see figure 5.2).

5.2 RPKI Adoption Benefits

As we mentioned in section 2.3.4, the RPKI adoption is very slow (i.e., around 20% of the total ASes on the Internet), especially on the Internet backbone (e.g., top 100 ASes according to AS-Rank). Thus, using BPHS, we tried to answer the following research question: *What are the benefits of RPKI ROV adoption on the Internet's backbone for reducing the Type-0 attacks?* Next, we present the methodology that we followed to answer the above question and the experimental results.

5.2.1 First Experiment: Methodology

1. The first step of the experiment was the collection of the top-500 ASes (i.e., customer cone, see section 2.4.2) using the API of the CAIDA AS-Rank service. The API is enabled to extract, in descending order, the AS-number of the top-X ASes, where X is the needed portion. The extracted list was used for tagging the corresponding ASes in simulation topology as ROV-enabled with probability p.

2. In the second step we defined the different simulation scenarios using the following assumptions:

   (a) Only the first X top 100 ASes perform ROV, with probability p.

   (b) All the other ASes of the topology do not perform ROV.

   Specifically, we considered different probabilities of adoption p (between 0.25 to 1), and different numbers of adopters X (between 0 to 100), chosen randomly from the set of $\frac{X}{p}$ top ISPs.

3. Finally, for each *{adoption probability, adopters portion}* combination we executed 20 random simulations from which we extracted the average attacker success rate (i.e., the average impact estimation after hijack). In total, we executed 1760 random simulations (prefix + sub-prefix hijacks) using the *static* ROV type of BPHS.

5.2.2 First Experiment: Results

In Figure 5.4, we see the experiment results for the different *{adoption probability, adopters portion}* combinations. The X-axis depicts the adopter's portion (between 0 to 100) and the Y-axis the attacker's success rate for each adopter portion - adoption probability pair (which is depicted with a unique color). We observe that, when the adoption portion and adoption probability increases, the attacker's success rate decreases constantly. Also, we observe that when the adoption probability is low (i.e., 0.25, as it is today on Internet's backbone) the attacker's success rate decreases slowly. We found that, when all the top 100 ASes apply ROV (probability 1.0), the attacker's success rate decreases around 30% for the prefix hijacks and around 40% for the sub-prefix hijacks.

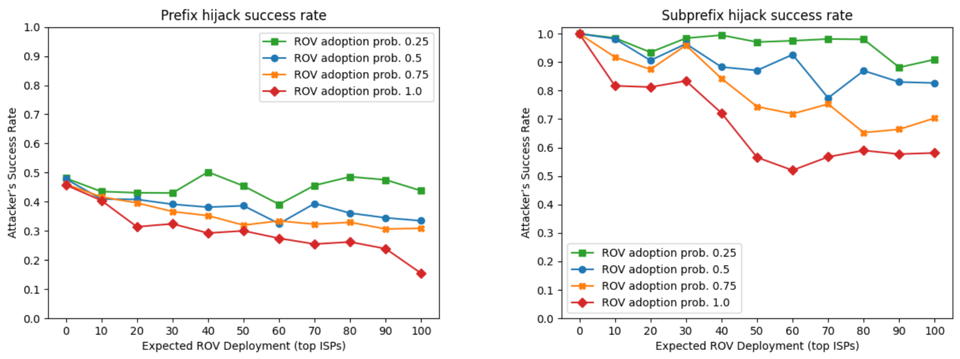


Figure 5.4: RPKI adoption benefits: Enforcing ROV only at top 100 ISPs reduces significantly the attacker's success rate.

5.2.3 Second Experiment: Methodology

1. The first step of the experiment was the collection of:

   (a) the top-100 ASes using the API of the CAIDA AS-Rank service.

   (b) the ASes that apply ROV according to the most up-to-date reported dataset extracted from the research results of Rodday *et al* [117].

   Specifically, we used the public available dataset of Rodday *et al* (see [45] 19-07-2021), including 206 unique ASes performing ROV in today's Internet of which 146 are fully and 60 are partially filtering (see section 4.2 *Different ROV modes* for more information).

2. In the second step, using the aforementioned datasets, we defined a comparative study including two simulation scenarios:

   (a) In the 1st scenario, we assumed as ROV-enabled ASes the top-100 and randomly selected ASes different from the top-100 with probability p.

   (b) In the 2nd scenario, we assumed as ROV-enabled ASes those reported in [45] and randomly selected ASes different from those in [45] with probability p.

   We clarify that the randomly selected BGP-nodes from the simulation topology are chosen from the set $\frac{totalNodes-X}{p}$, where $X = \{\text{top100}, 206\}$ and p in $\{0.0, 0.1, \dots ,1.0\}$ (we ignored the ASes of X that did not exist in the simulation topology).

3. Finally, for each probability p in $\{0.0, 0.1, \dots ,1.0\}$ we executed 20 random simulations from which extracted the average attacker success rate (i.e., the average impact estimation after hijack). In total, we executed 880 random simulations (prefix + sub-prefix hijacks) using the *static* ROV type of BPHS.

5.2.4 Second Experiment: Results

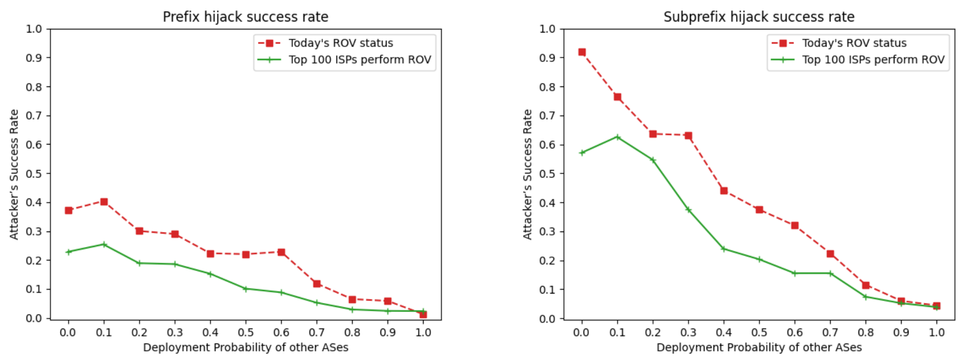


Figure 5.5: Comparing the benefits of RPKI under (a) today's ROV deployment, and (b) when the top 100 ISPs perform ROV for prefix and sub-prefix hijacks.

In Figure 5.5, we see the experiment results of the 2 scenarios (red and green line) for the different RPKI adoption probabilities p. The X-axis depicts the RPKI adoption probability (between 0 to 1) of the ASes that are different from the *top-100* or the *206* reported by Rodday *et al* [45]. The Y-axis depicts the attacker's success rate for each RPKI adoption probability. As we expected, the attacker's success rate reduces constantly when the adoption probability of the *other-ASes* increases (for both scenarios). Also, we observe better results for each adoption probability of the *other-ASes*, when we constantly perform ROV at the top-100 ISPs compared with the *today's status* (e.g., we see around 15% reduction for prefix hijacks and 40% reduction for sub-prefix hijacks with p=0).

5.3 Simulations Insights

In section 4.2, we presented two basic features of BPHS, which are the AS/Country vulnerability ranking for each simulation. During the evaluation phase (see section 5.1, 5.2) we executed up to 3000 simulations of random type using different ROV modes from which we extracted 2 ranking lists showing the most vulnerable ASes/Countries in descending order according to a ranking score. Next, we present the AS/Country ranking methodology and the corresponding results - insights.

5.3.1 AS/Country Vulnerability Ranking: Methodology

For the total AS vulnerability ranking, we collected the top 1000 vulnerable ASes of each executed simulation and we merged them in a dictionary having as keys the unique AS-number of each vulnerable AS and as values a list including the average vulnerability values of the AS (e.g., {"ASN": [0.5, 0.2, 0.8]} see section 4.2 for more information). Then, for each ASN in the dictionary, we counted the elements of the corresponding vulnerability list and we sorted the ASNs in descent order according to the resulting count values of all entries divided by the number of the executed simulations (e.g., for the above example the resulting value is {"ASN": $\frac{3}{3000}$ }). In Table 5.1, we see the computation formula of the AS vulnerability score.

$$\boxed{AS\_vulnerability\_score = \frac{\#AS\_appeared\_as\_infected}{\#total\_simulations}}$$

Table 5.1: Total AS vulnerability ranking score: computation formula.

For the total Country vulnerability ranking, we used the aforementioned AS vulnerability dictionary and the data provided by *ASN_ TO_ ORG* table to identify the country of each AS (see section 4.3.2). Specifically, we constructed a new dictionary setting as keys the 2-letter ISO code of each country and as values a list including the ranking score of the ASes belonging in this country. The ranking score of each country resulting from the average ranking score of its ASes and the number of its infected ASes in the total infected ASes. In Table 5.2, we see the computation formula of the Country vulnerability score.

$$Country\_vulnerability\_score = \frac{1}{n}\sum_{i=1}^{n} s_i + \frac{n}{t} = \frac{1}{n}(s_1 + \cdots + s_n) + \frac{n}{t}$$

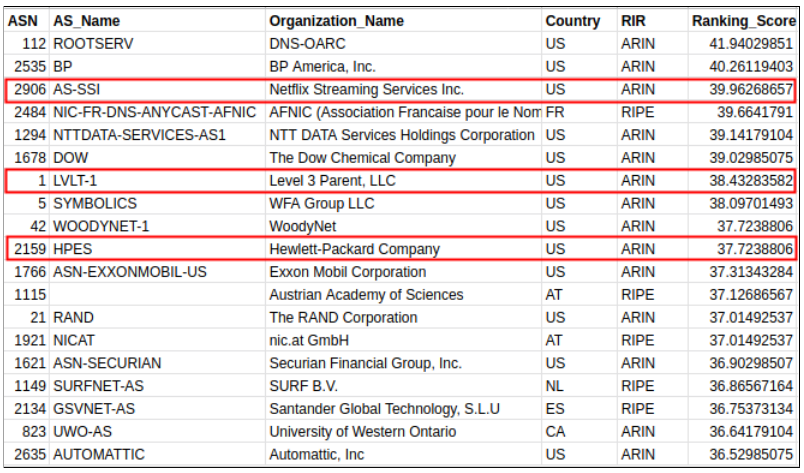
where:
n = #infected ASes belonging to the country,
s_i = the vulnerability score of the i AS
t = #total ASes

Table 5.2: Total Country vulnerability ranking score: computation formula.

5.3.2 AS/Country Vulnerability Ranking: Results

In Figures 5.6 and 5.7, we see the ranking results of the top 20 and top 100 vulnerable ASes respectively. The most vulnerable AS according to our simulations is called *"ROOTSERV"* (ASN: 112) which belongs to the DNS-OARC organization and is registered in ARIN RIR. *"ROOTSERV"* is a stub AS (has no customers) and has been ranked 47072 by CAIDA AS-Rank. The high vulnerability score of *"ROOTSERV"* is due to the high connectivity with backbone ASes (Tier1, Tier2 ASes as providers or peers) and thus, the probability to be hijacked is bigger than other ASes with fewer direct links to the backbone. In the second position, we meet the ASN 2535 belonging to the well known *"BP"* company which produces and retails oil and natural gas, and in the third position, we see the ASN 2906 belonging to the *"Netflix"* organization which is one of the most popular streaming services provider.

Surprisingly, in the 93rd position, we see the Greek ASN 2546 (*"ARIADNE-T"*) which is a stub AS having only one upstream link to *"GRNET"*. The *"GRNET"* is



ASN	AS_Name	Organization_Name	Country	RIR	Ranking_Score
112	ROOTSERV	DNS-OARC	US	ARIN	41.94029851
2535	BP	BP America, Inc.	US	ARIN	40.26119403
2906	AS-SSI	Netflix Streaming Services Inc.	US	ARIN	39.96268657
2484	NIC-FR-DNS-ANYCAST-AFNIC	AFNIC (Association Francaise pour le Nom	FR	RIPE	39.6641791
1294	NTTDATA-SERVICES-AS1	NTT DATA Services Holdings Corporation	US	ARIN	39.14179104
1678	DOW	The Dow Chemical Company	US	ARIN	39.02985075
1	LVLT-1	Level 3 Parent, LLC	US	ARIN	38.43283582
5	SYMBOLICS	WFA Group LLC	US	ARIN	38.09701493
42	WOODYNET-1	WoodyNet	US	ARIN	37.7238806
2159	HPES	Hewlett-Packard Company	US	ARIN	37.7238806
1766	ASN-EXXONMOBIL-US	Exxon Mobil Corporation	US	ARIN	37.31343284
1115		Austrian Academy of Sciences	AT	RIPE	37.12686567
21	RAND	The RAND Corporation	US	ARIN	37.01492537
1921	NICAT	nic.at GmbH	AT	RIPE	37.01492537
1621	ASN-SECURIAN	Securian Financial Group, Inc.	US	ARIN	36.90298507
1149	SURFNET-AS	SURF B.V.	NL	RIPE	36.86567164
2134	GSVNET-AS	Santander Global Technology, S.L.U	ES	RIPE	36.75373134
823	UWO-AS	University of Western Ontario	CA	ARIN	36.64179104
2635	AUTOMATTIC	Automattic, Inc	US	ARIN	36.52985075

Figure 5.6: Top 20 Vulnerable ASes.

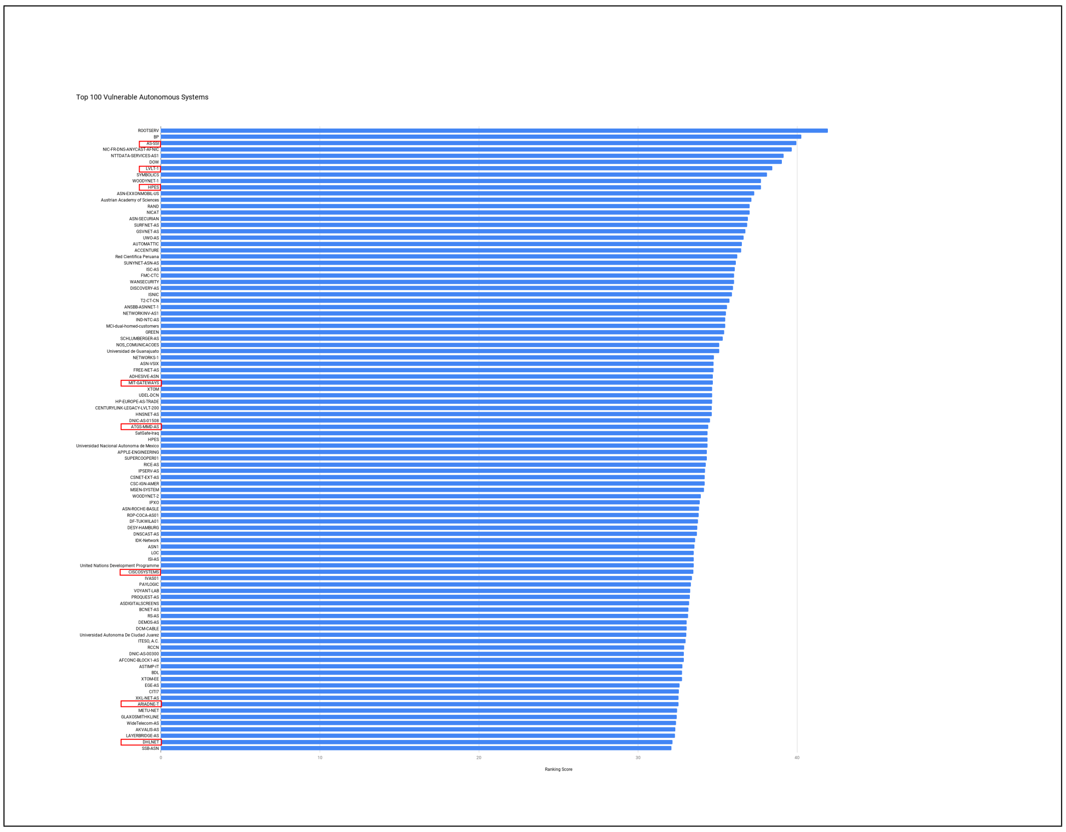
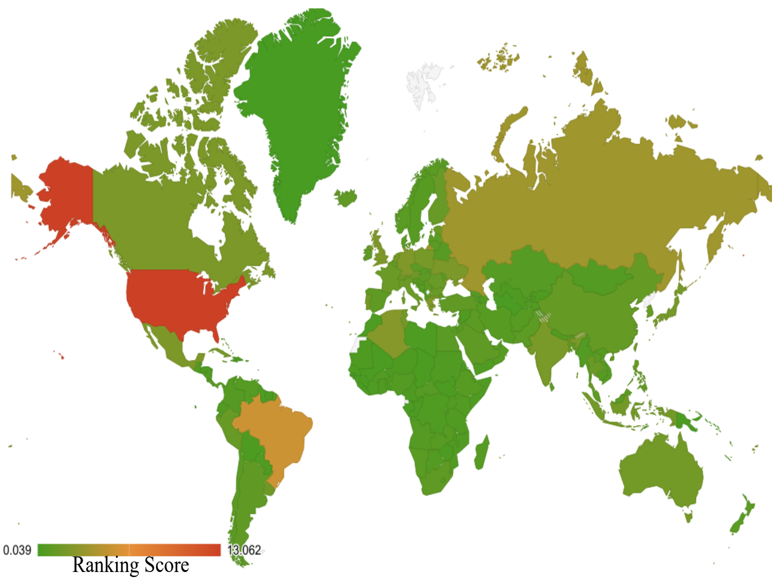


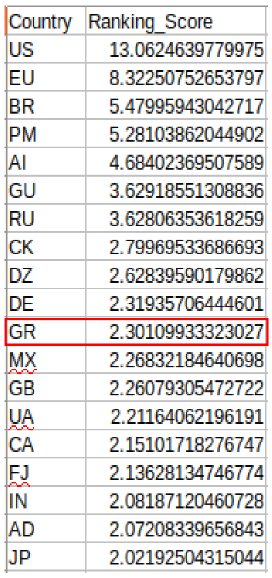
Figure 5.7: Top 100 Vulnerable ASes.

the Pan-Hellenic infrastructure for research and technology including the universities and school networks. *"GRNET"* has 2 upstream links to GEANT Vereniging (Tier2 AS), which is the European National Research and Education Networks (NRENs), and 1 upstream link to *"Cloudflare"* (Tier2 AS). *"NRENs"* and *"Cloudflare"* have a lot of upstream links to the biggest Tier1 ASes on the Internet's backbone (e.g., Level3, Telia Company AB, Cogent Communications, Hurricane Electric LLC, Internet2) and thus, it is more likely to propagate a malicious BGP route to their customers (e.g., to GRNET and GRNET to ARIADNE-T). In Table 5.3, we highlight the 7 most vulnerable ASes in Greece, according to the results of our simulations (we replaced "ARIADNE-T" with "GRNET" because when a stub AS is infected, its provider is also infected).

AS name	Ranking Position
GR-NET	93
FORTHNET-GR	526
HOL-GR	1366
HELEX-RP	2011
NATIONAL-BANK-OF-GREECE	2247
OTEGlobe	3573
SYNAPSECOM-AS	3633

Table 5.3: Top 7 vulnerable ASes in Greece.





Country	Ranking_Score
US	13.0624639779975
EU	8.32250752653797
BR	5.47995943042717
PM	5.28103862044902
AI	4.68402369507589
GU	3.62918551308836
RU	3.62806353618259
CK	2.79969533686693
DZ	2.62839590179862
DE	2.31935706444601
GR	2.30109933323027
MX	2.26832184640698
GB	2.26079305472722
UA	2.21164062196191
CA	2.15101718276747
FJ	2.13628134746774
IN	2.08187120460728
AD	2.07208339656843
JP	2.02192504315044

Figure 5.8: Country vulnerability Geo-chart.

In Figure 5.8, we see the Country vulnerability ranking results in a Geo-chart. Each country is colored according to its ranking score; higher vulnerability gives a color closer to red. As we expected, the most vulnerable country is the United States of America, due to a large number of ASes belonging to it. Brazil ranked in 3rd position and Russia ranked 7th. Surprisingly, Greece ranked 11th behind Germany, due to its high connectivity with the Internet's backbone.

5.4 Execution time and memory consumption

The execution time and the memory consumption of BPHS depend on the computational resources and power of the server that it is installed on, and also the number of Workers that have been defined by the administrator enabling the parallel execution of multiple simulations (see section 4.3 for more information). We measured BPHS's execution time and memory consumption on a desktop with Intel i7 4790 CPU, 32GB RAM, and 5 *Workers*. In Tables 5.4, 5.5, we present the evaluation results.

Number of Simulations	Time
1	21sec
20	7-8min

Table 5.4: Simulation execution time (5 *Workers*).

#Parallel Simulations	RAM
1	2GB
5	9-10GB

Table 5.5: Simulation memory requirements (5 *Workers*).

Chapter 6

Conclusions & Future Work

6.1 Conclusions

In this thesis, we implemented *BPHS*, the first BGP Prefix Hijacking Simulation tool that enables network operators to quickly and easily (a) assess the vulnerability of their Autonomous Systems to BGP prefix hijacks and (b) measure the benefits of the RPKI's adoption on the Internet, through a user-friendly web application. With BPHS, the network operators can simulate all the different types of BGP hijacking attacks and obtain the simulation results through an automated and graphical manner (i.e., well-designed Graphical User Interface).

The evaluation results showed that BPHS detects the real infected ASes of a prefix hijack with 70% mean accuracy and with 87% mean accuracy the infected ASes of a sub-prefix hijack. Also, using BPHS, we found that the RPKI filtering on Internet's backbone (top-100 ASes) reduces significantly the Type-0 attacks (30% for the prefix hijacks and 40% for the sub-prefix hijacks) even if all the other ASes in the Internet are not performing ROV. Finally, we showed that the most vulnerable ASes are those which have multiple Tier1 and Tier2 ASes as upstream providers (i.e., high connectivity in the Internet's backbone).

6.2 Future Work

Next, we elaborate on directions for future work:

- **Deployment on a public server.** Our future goal is to deploy BPHS on a public web server that will be accessible by anyone on the Internet, making it a public web *service* rather than a simple tool. The main challenge of this idea is to find the proper infrastructure to make the deployment. Ideally, we need a powerful web server that can handle a huge number of simulation requests and satisfy all the user's requests. A second challenge is the simulation limitations that should be addressed and the max users that can access our service concurrently to ensure stability and availability.

These limitations are very important to be applied, due to the computational resources limitations of the infrastructure in CPU and memory. A simple policy is to allow a maximum set of 30 users to access concurrently our server, a max of 10 simulation repetitions per submitted simulation, and in total 20 active simulations. The third challenge of the public deployment is the malicious attacks on the web app (e.g., DoS, DDoS attacks). The security concerns can be tackled by deploying BPHS behind a secure and up-to-date web server like NGINX [39] or APACHE [3]. We believe that the public deployment of BPHS will give visibility to the simulator and will help all user types to easily and quickly make their experiments.

- **More filtering rules.** BPHS applies the same inbound and outbound policies for all the BGP nodes in the simulation topology (see section 4.3.1 for more information), because we want to ensure graph convergence and routing stability. However, each Autonomous System on the Internet applies specific inbound and outbound policies for each neighboring AS. A promising idea would be to find the real inbound-outbound policies of each AS in the simulation graph and make a table in our database that stores all the AS-policies pairs. Then in the simulation topology initialization, we will load these policies in the AS-graph making BPHS a super-realistic BGP simulator. There are 2 main challenges with this approach. The first challenge is the collection of the real policies of an AS because most ASes do not make their policies publicly available. The second challenge is to ensure graph convergence and routing stability by applying the new policies on the AS-graph.

- **Execution time reduction.** In section 5.4, we showed that the execution time of a hijacking simulation with one repetition is about 21 seconds. Two good ideas could contribute to the execution time reduction. The first idea is to omit invalid paths according to the routing policies, for the specific hijacking scenario submitted by the user. Thus, we avoid needless memory allocations resulting in better performance. The second idea is to subtract the stub-ASes from the simulation AS-graph and infer if they are hijacked by mapping the information of their providers or peers (e.g., if the provider is infected the customer is also, because the malicious routes are always propagating to the customers). Thus, we avoid needless memory allocations resulting in better performance.

- **AS/Country Ranking Validation.** In the evaluation section 5.3, we presented two rankings for the most vulnerable ASes and Countries based on our simulation results. A very good idea would be to validate these rankings using real historical data collected by ISPs or network research community. We tried to find public datasets that could help us to validate our rankings with no success. We believe that this is a very interesting research question for future work that could be answered only with collaboration between ISPs and network researchers.

- **Authentication/Authorization system.** BPHS could be extended with an integrated authentication, authorization system. The idea is to provide a *BPHS-account* for each user. With this approach, each user can login to his/her profile and fetch all the simulation events launched by his/her. To support this idea we should create two new tables in our database. The first table will store all (username, password) tuples, and the second table all the authorized sessions that can access the pages and the database data. Also, we need to add a new column on the *bgp_ hijacking_ simulations* table containing the username that submits each simulation. The passwords should be stored in the database using a cryptographic hash algorithm like MD5 or SHA-256 [22]. Finally, a new login and sign-up page should be added, and also some modifications in the frontend and backend module should be done.

Bibliography

[1] A Universally Unique IDentifier (UUID) URN Namespace, 2005. https://datatracker.ietf.org/doc/html/rfc4122.

[2] Angular, Superheroic JavaScript MVW Framework, 2022. https://angularjs.org/.

[3] Apache Web-Server, 2022. https://httpd.apache.org/.

[4] Archipelago (Ark) Measurement Infrastructure, 2021. https://www.caida.org/projects/ark/.

[5] As to organizations mappings. https://publicdata.caida.org/datasets/as-organizations/. Accessed: 2022-1-16.

[6] ASRank (CAIDA). https://asrank.caida.org/.

[7] BGP AS PATH. https://www.catchpoint.com/network-admin-guide/bgp-attributes.

[8] BGP Best Path Selection Algorithm. https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html.

[9] BGP Local Preference Attribute. https://networklessons.com/bgp/how-to-configure-bgp-local-preference-attribute.

[10] BGP Messages. https://www.networkurge.com/2020/09/bgp-messages.html.

[11] CAIDA AS Relationship dataset, 2021. https://www.caida.org/catalog/datasets/as-relationships/.

[12] CAIDA HI^3 PaaS, 2021. https://dev.hicube.caida.org/feeds.

[13] Caida's bgp (hijacking) observatory. https://catalog.caida.org/details/media/2020_caidas_bgp_hijacking_observatory_kismet. Accessed: 2021-12-28.

[14] Center for Applied Internet Data Analysis (CAIDA). https://www.caida.org/.

[15] Chaos - BGP and Traffic Simulation for Evaluation of Internet Resiliency Systems, 2022. https://github.com/VolSec/chaos.

[16] Chinese ISP Hijacks the Internet. https://www.techtarget.com/searchsecurity/news/252452732/Google-BGP-route-leak-was-accidental-not-hijacking.

[17] Cisco Networking Academy's Introduction to Routing Dynamically. https://www.ciscopress.com/articles/article.asp?p=2180210&seqNum=12.

[18] CISCO Press - BGP Fundamentals - BGP Communities, 2021. https://www.ciscopress.com/articles/article.asp?p=2756480&seqNum=12.

[19] CISCO routers, BGP—Origin AS Validation. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/15-s/irg-15-s-book/irg-origin-as.pdf.

[20] Control Plane in routing. https://en.wikipedia.org/wiki/Control_plane.

[21] Cross-Origin Resource Sharing (CORS), Same-origin policy, 2022. https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy.

[22] Cryptographic hash algorithms, 2022. https://en.wikipedia.org/wiki/Cryptographic_hash_function#Cryptographic_hash_algorithms.

[23] Flask micro-framework. https://flask.palletsprojects.com/en/2.0.x/.

[24] Forward Plane in routing. https://en.wikipedia.org/wiki/Forwarding_plane.

[25] GNU Zebra: Free routing software distributed under GNU General Public License. http://www.zebra.org/.

[26] GR-IX Internet Exchange Point, 2021. https://www.gr-ix.gr/about/.

[27] Graphical Network Simulator 3 (GNS3), 2021. https://www.gns3.com/.

[28] Hacker Redirects Traffic From 19 Internet Providers to Steal Bitcoins. https://www.wired.com/2014/08/isp-bitcoin-theft/.

[29] Implementing Tunnels, Cisco, 2022. https://www.cisco.com/c/en/us/td/docs/ios/12_4/interface/configuration/guide/inb_tun.html.

[30] Internet Assigned Numbers Authority (IANA). https://www.iana.org/.

[31] Internet Exchange Point Looking Glass, 2021. https://www.uae-ix.net/en/resources/looking-glass.

[32] Internet exchange points dataset. https://catalog.caida.org/details/dataset/ixps. Accessed: 2022-1-8.

[33] JavaScript Object Notation (JSON), 2022. https://www.json.org/json-en.html.

[34] Jinja Template. https://jinja2docs.readthedocs.io/en/stable/.

[35] JUNIPER routers, Configuring Origin Validation for BGP. https://www.juniper.net/documentation/en_US/junos/topics/topic-map/bgp-origin-as-validation.html.

[36] Link aggregation, 2022. https://en.wikipedia.org/wiki/Link_aggregation.

[37] MPipe, Multiprocess Pipeline Toolkit for Python, 2022. https://vmlaker.github.io/mpipe/.

[38] Network Simulator 3 (GNS3), 2021. https://www.nsnam.org/.

[39] NGINX Web-Server, 2022. https://www.nginx.com/.

[40] NIST, "RPKI Monitor", 2021. https://rpki-monitor.antd.nist.gov/.

[41] Node Package Manager, 2022. https://www.npmjs.com/.

[42] Optimized BGP simulator to find paths between ASNs, 2022. https://github.com/Fraunhofer-SIT/bgpsim.

[43] PostgreSQL, The World's Most Advanced Open Source Relational Database, 2022. https://www.postgresql.org/.

[44] React, A JavaScript library for building user interfaces, 2022. https://reactjs.org/.

[45] Revisiting RPKI Route Origin Validation on the Data Plane: Datasets, Github Repository, 2022. https://github.com/nrodday/TMA-21/tree/main/data.

[46] RIPE Atlas Probes, 2022. https://atlas.ripe.net/about/probes/.

[47] RIPE Routing Information Service (RIS), 2021. https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris.

[48] RouteViews 6447, 2021. http://www.routeviews.org/routeviews/.

[49] Routinator, free open source RPKI Relying Party software, 2021. https://routinator.docs.nlnetlabs.nl/en/stable/.

[50] ROV Deployment Monitor, 2022. https://rov.rpki.net/.

[51] RSA (cryptosystem), 2022. https://en.wikipedia.org/wiki/RSA_(cryptosystem).

[52] Rsync, 2021. https://rsync.samba.org/.

[53] RTR protocol, 2021. https://blog.cloudflare.com/rpki-and-the-rtr-protocol/.

[54] RTRTR, RPKI data proxy, 2021. https://www.nlnetlabs.nl/projects/rpki/rtrtr/.

[55] Structured Query Language (SQL), 2022. https://en.wikipedia.org/wiki/SQL.

[56] The Internet Registry System. https://www.ripe.net/participate/internet-governance/internet-technical-community/the-rir-system.

[57] The Network Simulator NS-2, 2022. https://www.isi.edu/nsnam/ns/.

[58] Werkzeug (WSGI) web application library. https://werkzeug.palletsprojects.com/en/2.0.x/.

[59] WHOIS protocol, 2021. https://en.wikipedia.org/wiki/WHOIS.

[60] Robert Albrightson, JJ Garcia-Luna-Aceves, and Joanne Boyle. Eigrp–a fast routing protocol based on distance vectors. 1994.

[61] J Banks. Handbook of simulation: Wiley online library. 1998.

[62] S Bartholomew. The art of peering. *BT Technology Journal*, 18(3):33–39, 2000.

[63] Scott Bradner. The internet engineering task force (ietf). *DiBona et al.[144]*, pages 47–52, 1999.

[64] Markus Brandt and Haya Shulman. Optimized bgp simulator for evaluation of internet hijacks. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2. IEEE, 2021.

[65] Markus Brandt, Haya Shulman, and Michael Waidner. Evaluating resilience of domains in pki. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2444–2446, 2021.

[66] Tim Bruijnzeels, Oleg Muravskiy, Bryan Weber, and Rob Austein. The rpki repository delta protocol (rrdp). *IETF, Fremont, CA, USA, RFC*, 8182, 2017.

[67] RIPE Network Coordination Center. Youtube hijacking: A ripe ncc ris case study. *Amsterdam, The Netherlands (www. ripe. net/news/studyyoutube-hijacking. html)*, 2008.

[68] Jack Chan, Ray Chung, and Jack Huang. *Python API Development Fundamentals: Develop a full-stack web application with Python and Flask*. Packt Publishing Ltd, 2019.

[69] Ying-Ju Chi, Ricardo Oliveira, and Lixia Zhang. Cyclops: The internet as-level observatory. *ACM SIGCOMM Computer Communication Review*, 38(5):5–16, 2008.

[70] Taejoong Chung, Emile Aben, Tim Bruijnzeels, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, Roland van Rijswijk-Deij, John Rula, et al. Rpki is coming of age: A longitudinal study of rpki deployment and invalid route origins. In *Proceedings of the Internet Measurement Conference*, pages 406–419, 2019.

[71] David Clark. To filter or not to filter: Measuring the benefits of registering in the rpki today. In *Passive and Active Measurement: 21st International Conference, PAM 2020, Eugene, Oregon, USA, March 30-31, 2020, Proceedings*, volume 12048, page 71. Springer Nature, 2020.

[72] Avichai Cohen, Yossi Gilad, Amir Herzberg, and Michael Schapira. Jump-starting bgp security with path-end validation. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 342–355, 2016.

[73] Avichai Cohen, Yossi Gilad, Amir Herzberg, Michael Schapira, and Haya Shulman. Are we there yet? on rpki's deployment and security. 2017.

[74] Danny Cooper, Ethan Heilman, Kyle Brogle, Leonid Reyzin, and Sharon Goldberg. On the risk of misbehaving rpki authorities. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pages 1–7, 2013.

[75] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, KC Claffy, and George Riley. As relationships: Inference and validation. *ACM SIGCOMM Computer Communication Review*, 37(1):29–40, 2007.

[76] Xenofontas A Dimitropoulos and George F Riley. Efficient large-scale bgp simulations. *Computer Networks*, 50(12):2013–2027, 2006.

[77] Yuan Yu Michael Isard Dennis Fetterly, Mihai Budiu, Úlfar Erlingsson, and Pradeep Kumar Gunda Jon Currey. Dryadlinq: A system for general-purpose distributed data-parallel computing using a high-level language. *Proc. LSDS-IR*, 8, 2009.

[78] Behrouz A Forouzan. *TCP/IP protocol suite*. McGraw-Hill Higher Education, 2002.

[79] Agostino Funel. The graph structure of the internet at the autonomous systems level during ten years. *arXiv preprint arXiv:1902.05029*, 2019.

[80] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on networking*, 9(6):681–692, 2001.

[81] Veronica Gavrilă, Lidia Băjenaru, and Ciprian Dobre. Modern single page application architecture: a case study. *Studies in Informatics and Control*, 28(2):231–238, 2019.

[82] Tomas Hlavacek1 Italo Cunha23 Yossi Gilad and Amir Herzberg. Disco: Sidestepping rpki's deployment barriers.

[83] Yossi Gilad, Tomas Hlavacek, Amir Herzberg, Michael Schapira, and Haya Shulman. Perfect is the enemy of good: Setting realistic goals for bgp security. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 57–63, 2018.

[84] Yossi Gilad, Omar Sagga, and Sharon Goldberg. Maxlength considered harmful to the rpki. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 101–107, 2017.

[85] Phillipa Gill, Michael Schapira, and Sharon Goldberg. Let the market drive deployment: A strategy for transitioning to bgp security. *ACM SIGCOMM computer communication review*, 41(4):14–25, 2011.

[86] Phillipa Gill, Michael Schapira, and Sharon Goldberg. Modeling on quicksand: Dealing with the scarcity of ground truth in interdomain routing data. *ACM SIGCOMM Computer Communication Review*, 42(1):40–46, 2012.

[87] V. Giotsas, S. Zhou, M. Luckie, and k. claffy. Inferring multilateral peering. In *ACM SIGCOMM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 247–258, 2013-12.

[88] Vasileios Giotsas, Shi Zhou, Matthew Luckie, and Kc Claffy. Inferring multilateral peering. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 247–258, 2013.

[89] Timothy G Griffin and Gordon Wilfong. An analysis of bgp convergence properties. *ACM SIGCOMM Computer Communication Review*, 29(4):277–288, 1999.

[90] Suraj G Gupta, Mangesh M Ghonge, Parag D Thakare, and PM Jawandhiya. Open-source network simulation tools: An overview. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2(4):1629–1635, 2013.

[91] Susan Hares, Yakov Rekhter, Tony Li, and E Addresses. A border gateway protocol 4 (bgp-4). *nd, http://tools. ietf. org/html/rfc4271*, 2006.

[92] Ethan Heilman, Danny Cooper, Leonid Reyzin, and Sharon Goldberg. From the consent of the routed: Improving the transparency of the rpki. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 51–62, 2014.

[93] Tomas Hlavacek, Italo Cunha, Yossi Gilad, Amir Herzberg, Ethan Katz-Bassett, Michael Schapira, and Haya Shulman. Disco: Sidestepping rpki's deployment barriers. In *Network and Distributed System Security Symposium (NDSS)*, 2020.

[94] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. Practical experience: Methodologies for measuring route origin validation. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 634–641. IEEE, 2018.

[95] Geoff Huston and George Michaelson. Validation of route origination using the resource certificate public key infrastructure (pki) and route origin authorizations (roas), 2012.

[96] Daniele Iamartino, Cristel Pelsser, and Randy Bush. Measuring bgp route origin registration and validation. In *International Conference on Passive and Active Network Measurement*, pages 28–40. Springer, 2015.

[97] Raj Jain. *The art of computer systems performance analysis*. john wiley & sons, 2008.

[98] Cheng Jin, Qian Chen, and Sugih Jamin. Inet: Internet topology generator. 2000.

[99] Yuchen Jin, Colin Scott, Amogh Dhamdhere, Vasileios Giotsas, Arvind Krishnamurthy, and Scott Shenker. Stable and practical {AS} relationship inference with problink. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 581–598, 2019.

[100] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Autonomous security for autonomous systems. *Computer Networks*, 52(15):2908–2923, 2008.

[101] John Kristoff, Randy Bush, Chris Kanich, George Michaelson, Amreesh Phokeer, Thomas C Schmidt, and Matthias Wählisch. On measuring rpki relying parties. In *Proceedings of the ACM Internet Measurement Conference*, pages 484–491, 2020.

[102] Mohit Lad, Ricardo Oliveira, Beichuan Zhang, and Lixia Zhang. Understanding resiliency of internet topology against prefix hijack attacks. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pages 368–377. IEEE, 2007.

[103] Matt Lepinski and Stephen Kent. An infrastructure to support secure internet routing, 2012.

[104] Ziping Liu and Bidyut Gupta. Study of secured full-stack web development. In *CATA*, pages 317–324, 2019.

[105] M. Luckie, B. Huffaker, k. claffy, A. Dhamdhere, and V. Giotsas. As relationships, customer cones, and validation. In *ACM Internet Measurement Conference (IMC)*, pages 243–256, 2013-10.

[106] Gary Malkin. Rip version 2-carrying additional information. Technical report, RFC 1388, Xylogics, Inc, 1993.

[107] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces.* " O'Reilly Media, Inc.", 2011.

[108] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. Brite: An approach to universal topology generation. In *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 346–353. IEEE, 2001.

[109] Alexandros Milolidakis, Tobias Bühler, Marco Chiesa, Laurent Vanbever, and Stefano Vissicchio. Poster: Smart bgp hijacks that evade public route collectors.

[110] Asya Mitseva, Andriy Panchenko, and Thomas Engel. The state of affairs in bgp security: A survey of attacks and defenses. *Computer Communications*, 124:45–60, 2018.

[111] John T Moy. *OSPF: anatomy of an Internet routing protocol.* Addison-Wesley Professional, 1998.

[112] Ravi Musunuri and Jorge A Cobb. An overview of solutions to avoid persistent bgp divergence. *IEEE network*, 19(6):28–34, 2005.

[113] Bruno Quoitin and Steve Uhlig. Modeling the routing of an autonomous system with c-bgp. *IEEE network*, 19(6):12–19, 2005.

[114] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 291–302, 2006.

[115] Yakov Rekhter and Tony Li. Rfc1771: A border gateway protocol 4 (bgp-4), 1995.

[116] Andreas Reuter, Randy Bush, Italo Cunha, Ethan Katz-Bassett, Thomas C Schmidt, and Matthias Wählisch. Towards a rigorous methodology for measuring adoption of rpki route validation and filtering. *ACM SIGCOMM Computer Communication Review*, 48(1):19–27, 2018.

[117] Nils Rodday, Ítalo Cunha, Randy Bush, Ethan Katz-Bassett, Gabi Dreo Rodosek, Thomas C Schmidt, and Matthias Wählisch. Revisiting rpki route origin validation on the data plane. In *Proc. of Network Traffic Measurement and Analysis Conference (TMA). IFIP. accepted for publication*, 2021.

[118] Pavlos Sermpezis and Vasileios Kotronis. Inferring catchment in internet routing. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(2):1–31, 2019.

[119] Pavlos Sermpezis, Vasileios Kotronis, Konstantinos Arakadakis, and Athena Vakali. Estimating the impact of bgp prefix hijacking. *arXiv preprint arXiv:2105.02346*, 2021.

[120] Pavlos Sermpezis, Vasileios Kotronis, Petros Gigis, Xenofontas Dimitropoulos, Danilo Cicalese, Alistair King, and Alberto Dainotti. Artemis: Neutralizing bgp hijacking within a minute. *IEEE/ACM Transactions on Networking*, 26(6):2471–2486, 2018.

[121] Mohammad Sharif and Abolghasem Sadeghi-Niaraki. Ubiquitous sensor network simulation and emulation environments: A survey. *Journal of Network and Computer Applications*, 93:150–181, 2017.

[122] Vatika Sharma and Meenu Dave. Sql and nosql databases. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8), 2012.

[123] Saba Siraj, A Gupta, and Rinku Badgujar. Network simulation tools survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(4):199–206, 2012.

[124] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 618–627. IEEE, 2002.

[125] Pierre-Antoine Vervier, Olivier Thonnard, and Marc Dacier. Mind your blocks: On the stealthiness of malicious bgp hijacks. In *NDSS*, 2015.

[126] Matthias Wählisch, Robert Schmidt, Thomas C Schmidt, Olaf Maennel, Steve Uhlig, and Gareth Tyson. Ripki: The tragic story of rpki deployment in the web ecosystem. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, pages 1–7, 2015.

[127] Maciej Wojciechowski, Benno Overeinder, Guillaume Pierre, Maarten Van Steen, and Janina Mincer-daszkiewicz. Border gateway protocol modeling and simulation. 2008.

[128] Ellen W Zegura, Kenneth L Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE INFOCOM'96. Conference on Computer Communications*, volume 2, pages 594–602. IEEE, 1996.

[129] Man Zeng, Xiaohong Huang, Pei Zhang, and Dandan Li. Understanding the impact of outsourcing mitigation against bgp prefix hijacking. *Computer Networks*, 202:108650, 2022.