



SAPIENZA
UNIVERSITÀ DI ROMA

A practical emulator for BGP security

Facoltà di Ingegneria dell'informazione, informatica e statistica
Laurea Magistrale in Cybersecurity

Sara Bruzzese

ID number 1648786

Advisor

Prof. Francesca Cuomo

Co-Advisor

Dr. Flavio Luciani

Academic Year 2021/2022

Sara Bruzzese

A practical emulator for BGP security

Tesi di Laurea Magistrale. Sapienza University of Rome

© 2022 Sara Bruzzese. All rights reserved

This thesis has been typeset by L^AT_EX and the Sapthesis class.

Version: October 17, 2022

Author's email: bruzzese.1648786@studenti.uniroma1.it

*Dedicato a
me stessa.*

Abstract

Every day, the internet and its devices are attacked by hundreds of hackers from all over the world. One of the most difficult challenges of modern information technology is to secure our personal data and communications. The Border Gateway Protocol (BGP) is at the heart of how networks direct traffic across the Internet, and it offers the flexibility and scalability required to support Internet expansion. However, BGP, like many other protocols, was developed before Internet security was a prominent concern, and because BGP lacks built-in security safeguards, certain issues do arise. Nowadays, there are effective countermeasures for securing BGP sessions, but unfortunately they are still seldom used. I will demonstrate the value of having a sandbox environment where attacks can be reproduced without taking risks and damaging networks, with the possibility of studying the consequences and applying countermeasures to prevent future damages. Furthermore, I will demonstrate the incredible improvement provided by these security mechanisms, that can prevent and in some cases even nullify the effects of hacker attacks.

Acknowledgement

First of all, I would like to thank Professor Cuomo who believed in me and made my thesis work possible. I thank Flavio Luciani and Pietro Spadaccino for their brilliant and constant support during this work. I thank my mother and father for allowing me to study and for supporting me during these formative years. I thank Jessica for her infinite patience and for writing the index. I thank Eleonora for the bottarga pasta and Linda for her excellent advices. I thank my brother Davide for the nights spent together during the writing of this thesis. And I thank everyone who knows that have preciously contributed to my personal growth during this delicate period.

Contents

Introduction	1
1 Border Gateway Protocol	3
1.1 Historical Background	4
1.2 Definition of Autonomous System	5
1.3 Basic Functionality	6
1.4 BGP Session	7
1.5 BGP Messages	9
1.6 BGP Attributes	9
1.7 Best Path Selection	11
2 BGP Security	13
2.1 Vulnerabilities and Attacks	13
2.1.1 Prefix Hijack	14
2.1.2 Route Leak	15
2.1.3 Ip Spoofing	17
2.2 Countermeasures	17
2.2.1 Filtering IRR	18
2.2.2 RPKI	19
2.2.3 BGPsec	21
2.2.4 ASPA	22
2.3 Recent BGP Incidents	23
2.3.1 Security Bulletins and Statistics	23
2.3.2 Major Incidents Taxonomy	26
2.3.3 Example of Prefix Hijack	28
3 Twitter Prefix Hijack	31
3.1 Case Study Description	31
3.2 Incident Analysis	33
3.2.1 Analysis by BGPlay Tool	33
3.2.2 ASs Involved and Relationship Analysis	36
3.3 Selection of AS-subnetwork	40
4 Kathará Environment	41
4.1 Origins and Scope	41
4.2 Characteristics and Functioning	42
4.3 Application	44

5	KathBuilder Software Development	47
5.1	Input	48
5.2	Processing	49
5.3	Output	53
6	Experimental Laboratories	55
6.1	Initial State	57
6.2	Attack State	59
6.3	Countermeasure State	60
	Conclusion	63
A	Twitter’s Prefix Hijack Autonomous Systems Details	65
B	Results of CAIDA Script Execution	73
C	Results of Upstream Script Execution	83
	Bibliography	84

List of Figures

1.1	Internet flow graph example.	6
1.2	BGP functioning example.	6
1.3	Path Vector functioning example.	7
1.4	Classification of BGP Attributes.	10
2.1	Amount of prefix hijacks detected in the first half of 2020.	15
2.2	Route leak example.	16
2.3	AS64501 route object example.	19
2.4	ROA structure [10].	20
2.5	RPKI structure.	20
2.6	BGPsec signing chain.	22
2.7	BGP Hijack 2020 bulletin.	23
2.8	BGP Route Leaks 2020 bulletin.	24
2.9	Differences between 2019 and 2020 Security Bulletin.	24
2.10	BGP Hijack in 2021 bulletin.	24
2.11	BGP Route Leaks in 2021 bulletin.	25
2.12	Last 3 years security bulletin.	25
2.13	YouTube Prefix Hijack.	29
3.1	A portion of peers of AS8359.	31
3.2	DNS A records for twitter.com.	32
3.3	Malicious spread of Twitter's prefix.	32
3.4	BGPlay GUI during Twitter Prefix Hijack.	34
3.6	Progression of Twitter's prefix hijack with focus on AS8342.	35
3.7	Output of CAIDA script example execution.	38
3.8	Output of peval example execution.	38
3.9	Output of routeviews collectors example call.	38
3.10	Upstreams of AS1267 provided by bgpview.io.	39
3.11	Graph of selected Autonomous Systems.	40
4.1	Example of a network topology.	41
4.2	Docker container.	42
4.3	Example of lab.conf.	44
6.1	Kathará laboratories folder structure.	56
6.3	Basic configuration file examples.	56
6.5	Initial State diagram.	57

6.6	BGP table of AS21232 in the Initial State.	58
6.7	BGP update messages of AS13414.	58
6.8	Portion of IP routing table of AS13414.	59
6.9	Attack State diagram.	59
6.10	BGP table of AS21232 in the Attack State.	60
6.11	BGP table of AS8359 with RPKI off.	61
6.12	Advertised routes of AS8359 with RPKI off.	61
6.13	BGP table of AS8359 with RPKI on.	62
6.14	Advertised routes of AS8359 with RPKI on.	62

Listings

3.1	Python script that elaborates CAIDA dataset.	37
3.2	Python script that elaborates routeviews collector output.	39
5.1	Path elaboration function.	50
5.2	Peval elaboration module.	50
5.3	Lab.conf create function.	51
5.4	Configuration files create function.	52

Introduction

In recent years, we have witnessed an incredible development and growth of networks, technological devices and the application of these resources in our daily lives. More and more tasks are being performed by computers to simplify our jobs and our days. In some respects, technology has become indispensable. As always, when it comes to big opportunities and big changes, there are those who try hard to take advantage of the situation. The phenomenon of hacking has always gone hand in hand with the development of technology and its security, very often succeeding in overcoming it. The same fate has befallen the Internet environment, the backbone infrastructure for our instant connection and communication with the rest of the world. Almost every electronic device is now connected to the Internet to function, and this continually exposes them to dangers. Commonly, we are used to seeing the Internet from the outside as one big organism that works on its own, but it only takes a little digging to discover the great complexity behind every message sent over the web. The administration of the Internet is divided into large networks, called Autonomous Systems. Each of them is distinguished by technical IT elements, routing policies, administrative reasons. Internally, they are all managed autonomously and independently of each other, but they have one great point in common: the way they communicate. To date, the Border Gateway Protocol (BGP) is the most widely used protocol for communication and information exchange between Autonomous Systems. The protocol officially came into use in 1994, a period that was still not very mature in terms of security aspects. Thus, it was born and developed without internal security mechanisms, relying on trust and good network behaviour. As time went on, this lack proved to be significant, and several engineers and security experts had to patch a birth defect. From this history and premise, my interest in the BGP protocol was born, leading to it becoming my final work in my Master's degree in Cybersecurity.

In the first chapter of the thesis, I will discuss the basics of the Border Gateway Protocol, giving a brief historical overview and then moving on to explain its basic functionality, BGP sessions and the messages that are exchanged within them, how these messages can be manipulated to customise network use, and the most important mechanisms needed to understand the protocol.

In the second chapter, I will discuss its most famous vulnerabilities and how they are exploited to cause incidents and service interruptions. The most adopted countermeasures of today will also be discussed, and security bulletins and statistics will be reported to provide a general overview of the context, concluding with an analysis of a historical BGP incident that saw YouTube's network brought into trouble by Pakistan Telecom's network in 2008.

In the third chapter, I will discuss in detail the Twitter prefix hijack incident that occurred in March 2022, describing and analysing it through the use of various tools, venturing into the complex sphere of Autonomous System relationships and attempting to bring about a clear definition of them.

In the fourth chapter, I will introduce the framework on which my experimental thesis work is based. I will explain how it works, its characteristics, why it was chosen for this work and how its application.

In the fifth chapter, I will talk about the software I developed for this thesis work. I will explain why the need for it was felt, show snippets of code and explain how it works, presenting the output it provides.

In the sixth and final chapter, I will show the experiments and results I obtained using the KathBuilder software. The purpose of the experiments is to demonstrate the incredible effectiveness of BGP security countermeasures and to convince the reader of the usefulness of the KathBuilder software. Finally, the three appendices contain the detailed results of my studies, which I have separated from the body of the thesis for ease of reading.

Enjoy!

Chapter 1

Border Gateway Protocol

The BGP (Border Gateway Protocol) was developed as an EGP (Exterior Gateway Protocol) standard, designed to transmit routing information between Autonomous Systems. The version currently used is version number 4, defined in RFC 1771 in March 1995, rewritten with the same title in January 2006 as RFC 4271.

Its main characteristics are the following:

- is a Path Vector type routing protocol, that is, hops are measured in terms of the number of AS
- supports CIDR
- selects the best paths through a fairly complex selection procedure based on several metrics
- due to the variety of several types of metrics, it is possible to implement routing policies for both outgoing and incoming traffic in the AS
- TCP connections are employed to secure the transmission of routing information
- updates only when triggered

Discussing about the BGP protocol, it is very important to comprehend how BGP sessions are realized, what rules govern them, what messages the protocol employs to communicate routing information, the BGP attributes involved in the selection process, and how the selection process itself operates.

In the following paragraphs, I will describe the main notions that need to be known in order to understand how the protocol works.

1.1 Historical Background

In the early days of the Internet, what is known today as the 'network of networks' actually consisted of a single network: ARPANET, developed in the late 1960s, and its satellite extension SATNET, developed in the mid-1970s. The routers, known as gateways at the moment, communicated routing information using a single Distance Vector protocol known as GGP (Gateway-to-Gateway Protocol), which subsequently evolved into RIP (Routing Information Protocol), which remained the Internet's primary routing protocol for many years. However, as the number of users and nodes increased, it became clear that using a model without any form of hierarchy was not scalable at all: a single routing protocol was insufficient to handle the network's complexity. Thus, arose at this time the necessity for a hierarchical structure, splitting the Internet into a group of Autonomous Systems (AS), that is networks governed by the same administration. In January 1989, Yakov Rekhter and Kirk Loughheed, collaborated and set the foundations for a new inter-AS routing protocol: Border Gateway Protocol (BGP). Within a short period of time, they developed the first actual implementation of BGP, and therefore the first standard, were produced, as described in RFC 1105 - A Border Gateway Protocol (BGP), June 1989. The standardization process included several steps that led to the definition of a second and third version, which were published in RFC 1163 in July 1990 and RFC 1267 in October 1991, respectively. BGP version 4 is the de facto standard universally used as an inter-domain routing protocol. It has been subjected to continuous updates over the years which have enriched its functionality, stability and scalability. [39]

Over time, its functionality has been extended and nowadays it is used for:

- in the modern public networks of major ISPs (Internet Service Providers), where it plays an important part in the entire routing architecture
- in the control plan for VPN services
- as a private network access protocol to ISP networks

Today, we may say that BGP is the most essential routing information exchange protocol used by IP networks. In the modern computer age, it must be recognised that BGP was born with a defect: it assumes that all networks of which the Internet is composed are trusted. Unfortunately, especially in the last 30 years, this characteristic has endangered the security of networks, which have had to adapt over time to the emergence of new threats.

1.2 Definition of Autonomous System

An Autonomous System (AS) is a group of routers that are managed by a single organization and often use a single internal IGP protocol. Externally, an AS is considered a single entity, identified by a number encoded with 16 or 32 bits and assigned by the various Regional Internet Registries (RIRs): RIPE (Europe, West Asia, and the former Soviet Union), APNIC (Asia-Pacific Area: Central Asia, South-East Asia, Indochina, and Oceania), ARIN (North America, Atlantic Islands), LACNIC (Central-South America, Caribbean), AfriNIC (Africa (Africa)). Routing information is exchanged between ASs using protocols from the EGP (Exterior Gateway Protocol) family, which currently comprises only the BGP protocol.

AS are classified based on their external connectivity and transit traffic processing:

- **Single-homed ASs** - defined by a single or redundant connection to one AS
- **Multi-homed ASs** - defined by numerous connections to multiple ASs

A **single-homed AS** is defined by a single connection, or, more commonly, a redundant connection, to the same AS. In the case of a single connection, this is known as a *stub AS*. A stub AS with a single connection to the ISP does not need to learn all of the Internet prefixes. In practice, because the AS has a single point of contact with the outside world, it is reasonable to ensure reachability of prefixes outside the AS using a simple default route on the ISP's network access router.

Multi-homed ASs are made up of private networks or small ISPs that link to the networks of two or more public ISPs for reliability reasons.

Two types of multi-homed AS can be distinguished:

- *Multi-homed transit ASs* - they allow traffic to be exchanged between various ASs while using their own resources
- *Non-transit multi-homed AS* - do not allow transit traffic on the AS

A multi-homed AS becomes transit when it propagates BGP prefix announcements received from other ASs.

1.3 Basic Functionality

The whole Internet may be represented as a flow graph, with the nodes representing the AS and the links between them representing BGP sessions.

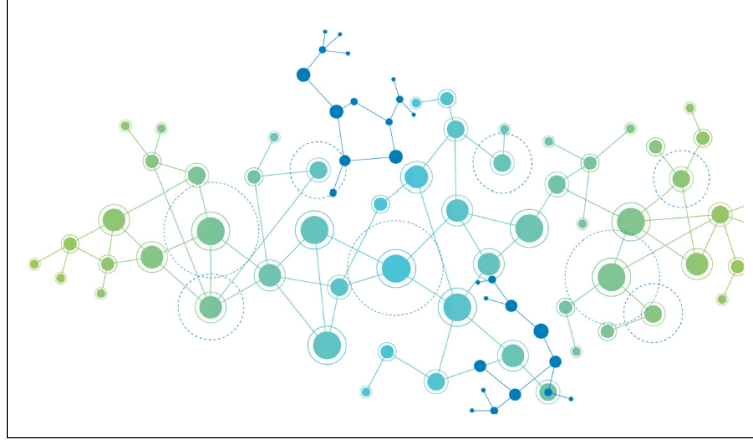


Figure 1.1. Internet flow graph example.

Routing information consists of pairs of the type:

$$\langle IPaddress, prefixlength \rangle$$

This pair is commonly referred to as the IP prefix.

Prefixes announced locally within an AS are automatically propagated over the various BGP sessions to create connections in the system. For the correct functioning of the Internet system, It is not essential to make all IP prefixes known to all routers on the Internet, which prefixes are propagated and to whom is mostly determined by the topology of the AS and the role they perform. The basic functioning of BGP consists of establishing sessions among routers, sending them their routing table containing the prefixes that each router is able to reach, associating with each prefix a distance measured in terms of autonomous systems to be traversed to reach the AS originating the prefix. On BGP sessions, routing information is exchanged via special protocol messages called UPDATE. Through UPDATE messages, it is also possible to inform other routers to withdraw previously sent prefix announcements. The following figure shows an example of its functioning.

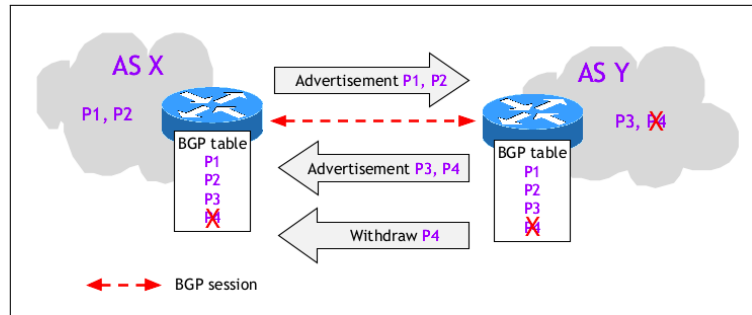


Figure 1.2. BGP functioning example.

The BGP protocol is of the Path Vector type, which means that the path that a packet must follow to reach a given prefix is indicated in terms of AS (*as-path*). For example, in the figure below, the prefix 192.0.2/24, belonging to AS64503, will be announced to AS64501 by both AS64502 and AS64505. In the former announcement the Path Vector is [64502 64503], while in the latter it is [64505 64504 64503]. As we will see below, the Path Vector plays a fundamental role in the functioning of the BGP.

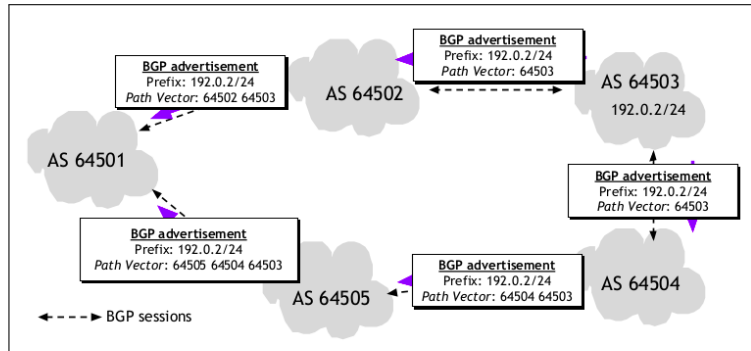


Figure 1.3. Path Vector functioning example.

In the next section, we look in detail at the functionality of BGP sessions, one of the main elements on which the protocol is based.

1.4 BGP Session

A BGP session is a reliable communication channel established between two routers used to exchange BGP messages, which uses a TCP connection on well-known port 179. Routers in which the BGP protocol is running are called *BGP speakers* (since they can speak the language of the protocol), while routers involved in a BGP session (and therefore connected to each other) are called *BGP Neighbours*.

BGP sessions can be of two types:

- external BGP (*eBGP*) - established between routers belonging to different ASs
- internal BGP sessions (*iBGP*) - established between routers belonging to the same AS

A significant difference between them is in the way announcements are forwarded. *eBGP* sessions adopt the following basic rule to propagate BGP announcements:

Each BGP announcement received by a router on an eBGP session, if becomes the best path, is automatically propagated on all active BGP sessions (eBGP and iBGP), except the one from which the announcement was received.

Instead, iBGP sessions adopt the following basic rule to propagate BGP announcements:

Each BGP advertisement received by a router from an iBGP session, if becomes the best path, is automatically propagated on all active eBGP sessions, but not on iBGP sessions.

The establishment of a BGP session consists of two basic steps: establishing the TCP connection on well-known port 179 and initialising the session.

The procedures for opening the TCP connection can be started by both routers at the extremes and typically start after a manual configuration. One of the two routers plays the active role with respect to opening the connection and initiates the classic three-way-handshake.

Once the TCP connection has been established, the next step is the initialisation of the BGP session, during which the two BGP Neighbours exchange the *BGP OPEN* message. This message is also used to negotiate parameters to improve the functionality of the connection and to perform verification checks. If there are no errors, the session is established. Typically, after the initial exchange of the best paths of the prefixes in the BGP table, no messages are exchanged between the two routers unless the BGP table itself changes. To keep the session open, empty BGP messages called *KEEPALIVE* are exchanged periodically.

The opening of a BGP session passes through 5 states, which are:

- **Idle** - the starting state, in which a BGP Speaker attempts to initiate a TCP connection to a BGP Neighbour and wait to move on to Connect state
- **Connect** - the BGP Speaker waits for TCP segments to complete the TCP connection, in case of success move on to the OpenSent state, otherwise move on to the Active state
- **Active** - the BGP Speaker attempts to re-open the TCP connection with the BGP Neighbour, in case of success move on to the OpenSent state, otherwise the connection is rejected and there's no change of state
- **OpenSent** - the BGP Speaker has already sent the OPEN message and waits for the response, then in case of success move on to OpenConfirm state, otherwise move on to Idle state
- **OpenConfirm** - the BGP Speaker waits for a KEEPALIVE or NOTIFICATION message communicating possible error conditions, in case of success move on to Established state, otherwise move on to Idle state
- **Established** - communication between the two BGP Neighbours is fully established and they can start exchanging routing information and possibly other BGP messages, in case of errors move on to Idle state

1.5 BGP Messages

BGP messages are transmitted by TCP/IP packets over a connection established on TCP port 179 either as source or destination. The length of BGP messages can vary from a minimum of 19 to a maximum of 4,096 bytes for OPEN and KEEPALIVE and 65,535 bytes for other messages. All messages have a common header, consisting of Marker, Length and Type.

Five types of messages are currently defined:

- **OPEN** - the first BGP message exchanged between two BGP Neighbours, immediately after activating the TCP connection and contains information such as the version of the protocol used and the number of the AS generating the message
- **UPDATE** - the most important BGP message and is used to announce IPv4 prefixes with associated BGP attributes and/or withdraw previously announced IPv4 prefixes
- **NOTIFICATION** - used to notify any error conditions
- **KEEPALIVE** - empty message exchanged periodically to keep the session open
- **ROUTE REFRESH** - used to properly set the inbound filters re-sending the best-path on BGP sessions

1.6 BGP Attributes

BGP attributes are the fundamental tool for the functioning of the entire protocol and for the implementation of routing policies, as they have a crucial effect on the BGP selection process of best paths.

A first classification divides the BGP attributes into two main classes:

- *Well Known* - attributes that must necessarily be recognised by any BGP implementation
- *Optional* - attributes that may not be recognised by a BGP implementation

The Well Known attributes are themselves divided into:

- *Well Known Mandatory* - these are mandatory attributes, that is, they must be present in all UPDATE messages
 - ORIGIN, AS_PATH, NEXT_HOP
- *Well Known Discretionary* - they don't need to be present in all announcements BGP
 - LOCAL_PREF, ATOMIC_AGGREGATE

The Optional attributes are themselves divided into:

- *Optional Transitive* - If the attribute is not recognised by the BGP implementation, it must ignore it and still transmit it to its BGP Neighbor
 - AGGREGATOR, COMMUNITY, EXTENDED_COMMUNITY, LARGE COMMUNITY, AS4_PATH, AS4_AGGREGATOR
- *Optional Non Transitive* - If the attribute is not recognised by the BGP implementation, it must ignore it and not transmit it to its BGP Neighbours.
 - MULTI_EXIT_DISC, AIGP, ORIGINATOR_ID, CLUSTER_LIST, MP_REACH_NLRI, MP_UNREACH_NLRI

The diagram is summarised in the figure below.

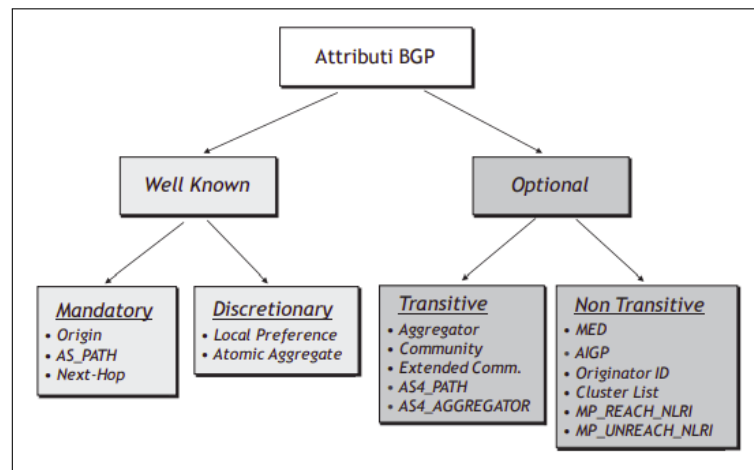


Figure 1.4. Classification of BGP Attributes.

In particular, the characteristics of some of the most used attributes:

- *ORIGIN* - specifies the origin of an IP prefix
- *AS_PATH* - specifies the list (ordered or unordered) of ASs traversed by a BGP announcement
- *NEXT_HOP* - specifies the next step to reach the prefixes contained in the destination field
- *LOCAL_PREF* - specifies a metric that can be used to assign a degree of preference to an announcement
- *ATOMIC_AGGREGATE* - used to notify to other BGP speaker that the router transmitting the announcement has performed a prefix aggregation operation with loss of memory of BGP attributes

- *COMMUNITY* - used to classify BGP announcements according to administrative needs and eventually execute actions on the basis of the attribute value
- *MULTI_EXIT_DISC (MED)* - specifies a metric that can be used in the best-path selection process

1.7 Best Path Selection

Normally, a router receives announcements of the same prefix from several sources. The announcements of all exchanged prefixes are stored in specific memory areas associated with the individual BGP sessions, and then subjected to possible manipulation using BGP attributes and/or by applying filtering policies. The BGP process within each router then chooses the best path from all the announcements of the same prefix and propagates it to the BGP sessions, following precise rules according to the type of protocol employed. The best path is selected on the basis of a well-ordered sequence of choices based on various metrics or announcement properties. This ordered sequence of choices is called the selection process, and produces a single best path.

The selection process is conducted in the following order:

1. Prefer announcements with the highest Local Preference value
2. Prefer announcements with the minimum number of elements in the AS_PATH attribute
3. Prefer announcements with the lowest value in the ORIGIN attribute
4. If the announcements come from the same AS, prefer the one with the lowest MED value
5. Prefer announcements from eBGP sessions than those from iBGP sessions
6. Prefer announcements that have the Next-Hop "closest" according to the total IGP cost
7. Prefer announcement from from the BGP peer with lower BGP Identifier
8. Prefer the announcement coming from the BGP peer with the lowest BGP Neighbour Address

The result of this process inevitably returns a single element, which leads to the final choice of the best path.

Chapter 2

BGP Security

In recent years, in order to cope with the significant transformations in the use of the Internet and its users, it became necessary to study and implement security measures for all routing protocols. This is especially true for the BGP protocol, since it's responsible for the connection of the entire Internet, and also because the exchange of BGP routing information involves a variety of entities on which it is impossible to make assumptions of complete reliability and ethical behaviour, as may have been possible in the past. When the protocol was created, no security features were included, because the technological context of the time did not required this need. Over the years, a number of operational mechanisms have been introduced, such as prefix filtering, and they can be thought as tools to help build a more secure BGP architecture. In this chapter, I will explain the main issues concerning BGP security, which are useful for the correct configuration and administration of network equipment. Finally, it should be clarified that the techniques presented in this chapter relate to eBGP sessions, where the counterpart is to be considered "untrusted", since it's administered by a different entity.

2.1 Vulnerabilities and Attacks

The entire Internet is vulnerable to attacks on its routing protocols, and BGP being one of the most important for its functioning, it has been widely targeted.

The areas on which attention should be focused can be summarised as follows:

- *Authentication* - verifying the identity of BGP Neighbor
- *Integrity* - verifying that BGP messages have not been illegally manipulated
- *Availability* - protecting a BGP Speaker from attacks aimed at overloading its processing resources and rendering it unusable
- *Origin Validation* - implementing mechanisms to verify whether or not an AS is authorised to announce a given IP prefix in the Internet ecosystem
- *AS_PATH Validation* - ensuring that the AS_PATH has not been manipulated from the source to the destination AS

The modes of attack are numerous, and a simple taxonomy defines two major types:

- *Session-level attacks* - may include attempts to modify the flow of BGP messages, such as message modification, insertion, and deletion, attempts to break down the session between two BGP neighbors, and attempts to extract secret information from BGP message interception. This section also includes all TCP attacks.
- *Denial of Service attacks* - these are attacks that tend to block a service from well operating, for example, by illegally terminating BGP sessions or, more commonly, by overloading a router's resources or even the capacity of transmission channel.

There is also unethical behavior on the part of network administrators that might be considered attacks. BGP is a protocol with a lot of metrics and capabilities, and unfortunately, network managers may abuse these features for malicious purposes. A common unethical practice consists in using an AS as transit without its consent, allowing for serious bandwidth theft.

Before describing specifically the most common types of attacks that I have analysed during this research, it is important to talk about why these attacks are possible. Below are the 3 most important vulnerabilities of the BGP protocol:

- BGP lacks means for ensuring the integrity and freshness of messages as well as authenticating their origin. The integrity of the message guarantees that it has not been altered, freshness assures that the receiver has gotten a new message and not one that has been reproduced by a man-in-the-middle, and authentication of the origin ensures that whomever originated the message is authorized to do so.
- The BGP protocol provides no mechanism for verifying ASs to guarantee that they are not authorized to distribute erroneous routing information. Any prefix, even if it does not belong to its AS, can be generate by an AS by illegally modifying specific BGP properties. The BGP performs no checks on the legality of these activities.
- The BGP has no way of verifying the authenticity of the AS_PATH attribute announced by a certain AS. Modifying the AS_PATH is one of numerous ways available to an unethical administrator to compromise or manipulate the routing architecture.

2.1.1 Prefix Hijack

One of the most common routing incident is caused by prefix hijack. BGP is used to announce a prefix, which includes an IPV4 or IPV6 address block as well as a route of AS numbers indicating which ASNs traffic must transit through to reach the announced address block. An attacker can redirect traffic in order to intercept or manipulate traffic by maliciously modifying BGP IP prefixes. These attack exploit

BGP's fundamental weakness: it cannot verify whether AS are legally authorized to send announcements. Prefix hijacking occurs when a network, whether on purpose or unintentionally, originates a prefix that belongs to another network without authorization. If this malicious announcement is accepted by nearby networks and spread further through BGP, it alters the Internet's roadmap. As a result, traffic is sent to the attacker rather than the intended destination. This is occasionally made by mistake, but it is more typically done to intentionally intercept communications and generate denial-of-service incidents. Below is a figure depicting the frequency of prefix hijacks detected between January and July 2020, with an average of 14 incidents per day. [40]

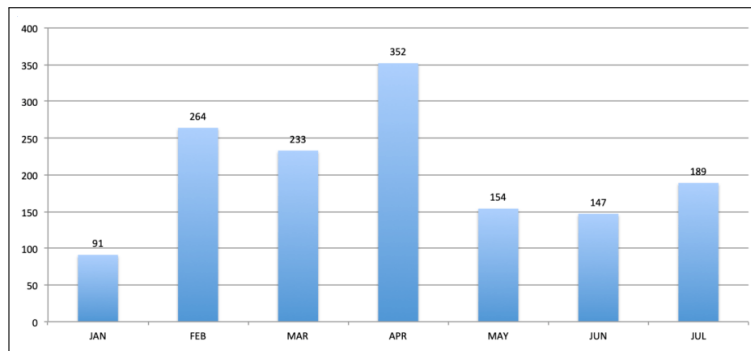


Figure 2.1. Amount of prefix hijacks detected in the first half of 2020.

There are a variety of techniques that can be employed to perform prefix hijacking, all of which aim to make their own malicious announcement the best-path for other routers. In this way, the attack will be successful and traffic will be redirected. Very common techniques are those exploiting the shortest as-path or the longest prefix match. The only way to prevent prefix hijacking issues is to build up an authorisation system that, through the use of digital certificates, can check whether or not an AS is authorized to announce a certain prefix.

2.1.2 Route Leak

To improve reliability or performance, several companies link to multiple networks or upstream providers, a process known as multihoming. When an organization inadvertently communicates to one upstream provider that it has a route to a destination via another upstream provider, regardless of whether this is a desirable path, this is referred to as a route leak.

As a result, the organization's service operates as an unintentional intermediary between the two upstream providers, with traffic flowing via its much smaller network. As a result, non-optimal routing, congestion, and probable traffic non-delivery occur. While leaks are frequently caused by inadvertent router misconfigurations that are quickly corrected, they can also be done on purpose to divert traffic through another network to scan it or to perform man-in-the-middle attacks, in which an attacker

secretly retransmits and manipulates communication between a sender and receiver. [18]

The technical definition for a route leak is provided by RF 7908

A route leak is the propagation of routing announcement(s) beyond their intended scope. That is, an announcement from an Autonomous System (AS) of a learned BGP route to another AS is in violation of the intended policies of the receiver, the sender, and/or one of the ASes along the preceding AS path.” [16]

The scope is specified by BGP import and export policies, which are used by ASs to manage the set of routes exchanged during a BGP session.

Typically, these rules are imposed by business agreements between ASs. They may be divided into two categories: provider-to-customer and peer-to-peer. In a provider-to-customer connection, the provider declares the routes to all internet destinations to the customer, while the customer announces to the provider the routes to its networks and the networks of its customers. In contrast, in a peer-to-peer connection, the two ASs transmit routes to their own networks and those of their clients. A leak occurs when an AS announces a route obtained from another provider or peer to another provider or peer. [5]

In the route leaks attacks, the consumer is basically transformed into a transit provider. This incident opens the door to a staged man-in-the-middle (MITM) attack or a Denial of Service (DoS) attack.

The figure below depicts the diverted flow via a downstream, allowing it to analyze traffic that is not its own.

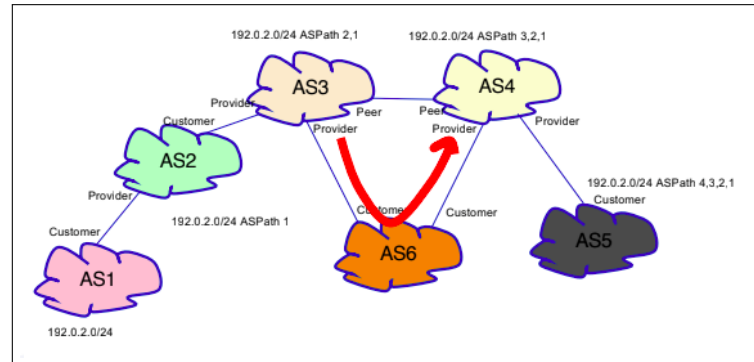


Figure 2.2. Route leak example.

Actually, there are few security solutions established to defend against this sort of attack, primarily because BGP lacks the capacity to communicate relationships such as customer-provider. [9]

2.1.3 Ip Spoofing

One of the most dangerous Internet attacks is BGP spoofing because it affects one of the internet's most fundamental functions: data routing from one system to another. The internet is a network made up of other networks. Routers are used to transfer data between networks using IP addresses recorded in their routing tables. Routers will communicate to each other that they use specific addresses. However, and this is critical, there is no authority to authenticate that a certain address belongs to a specific network. There are organizations that allocate IP addresses, such as RIPE in Europe and ARIN in the US and Canada, but there is no authoritative place to verify to validate such allocation. As a result, updating routing tables is totally based on trust. [19]

The data is transmitted across the Internet in packets that include the destination IP address. Packets also include a source IP address, which identifies the sender and allows the receiver to respond. Unfortunately, it is quite simple to build and send an IP packet with a fake, or spoofed, IP address in order to mask the sender's identity or impersonate another computing system. Because it might be difficult for routers to validate both destination and source IP addresses, they frequently do not bother and just forward them. [18]

The technique is so simple, nearly trivial, that tens of thousands of attacks based on impersonating other people's IP addresses have been detected in recent years. CAIDA has been conducting this study for many years, due in part to software called Spoofer, which can detect whether an autonomous machine sends out IP packets from a faked source. It is feasible to identify an attack in progress, but it is impossible to prevent or interrupt one that is already ongoing.

2.2 Countermeasures

As explained in previous paragraphs, the challenge with BGP is that the protocol does not directly include security mechanisms and is based largely on trust between network operators. Security protocol extensions are needed to prevent malicious use of the system.

Before explaining the most common and effective countermeasures used for BGP security, it is important to emphasise three key points:

- Security mechanisms do not solve the threat problem, but they can greatly mitigate it by trying to reduce the impact attacks have on the ecosystem as much as possible.
- Security mechanisms must be applied in combination and not individually, as this would only control one aspect while leaving other vulnerabilities uncovered.
- Operators' awareness of adopting these mechanisms must increase; if everyone used these mechanisms, the risk of threats would be significantly reduced.

2.2.1 Filtering IRR

One of the primary reasons for BGP's success as a network routing protocol is the ability to implement extremely flexible routing policies that satisfy nearly all of the demands of network managers. A routing policy defines an AS's rules for managing incoming and outgoing traffic, as well as the rules for receiving and transmitting BGP announcements.

The tools for defining a routing policy are essentially the filtering of announcements and the manipulation of BGP attributes. Filtering consists in preventing the propagation of incorrect routing information.

To achieve a proper standard of filtering, it is necessary that the network operator establishes a clear routing strategy and implements a system that verifies the validity of their own announcements as well as announcements from their customers to adjacent networks at the prefix and AS-path level. In addition, the operator has to confirm the accuracy of its customers' announcements, especially ensuring that the client legitimately owns the ASN and address space announced. It's very important to use explicit prefix-level filters or comparable mechanisms to safeguard inbound routing announcements, particularly from customer networks.

Moreover, AS-path filters might be used to enforce the customer network to specify which Autonomous Systems downstream they have. AS-path filters that exclude announcements by customers of ASes with which the provider has not a contract relationship can help to avoid various kinds of routing leaks. One way to implement filtering is using IRR and require the customers to register route objects.

Internet Routing Registries are important repositories for the publication of routing information. They specify which ASNs are permitted to announce which IP addresses, as well as the policies for exchanging routes between ASNs. This information may be used to validate received routes in router configurations. The AFRINIC, APNIC, and RIPE NCC IRRs permit users to store resource records that correspond to the resources they own. Other IRRs may have less stringent authentication methods in place to prevent users from providing false data.

To begin filter implementation, an operator will request that their customers register their intended announcements as route objects in a certain IRR.

A *route object* is an IRR element that specifies the Autonomous System Number that will propagate a certain IP prefix to the Internet. It can also be used to provide information about the company that owns this prefix. They are normally password-protected, and only the authorized user may modify them. Route objects are used by Internet Service Providers and Upstreams to authenticate the legitimacy of routes received from peers. When implemented by all providers, this typically helps to minimize route hijacking. [1]

The figure below shows an example of a route object registered by AS64501 and requested by its upstream.

```
route:      192.0.2.0/24
descr:      Cust 64501
origin:      AS64501
mnt-by:      MAINT-AS64501
created:     2015-09-27T12:14:23Z
last-modified: 2015-09-27T12:14:23Z
source:      RIPE

route6:      2001:db8:1001::/48
descr:      Cust 64501
origin:      AS64501
mnt-by:      MAINT-AS64501
created:     2015-09-27T12:14:23Z
last-modified: 2015-09-27T12:14:23Z
source:      RIPE
```

Figure 2.3. AS64501 route object example.

After the implementation of all required route objects, a smart tool will read the aut-num object that defines the upstream policy, gather all referenced objects, extract customer prefixes, and generate the proper ingress and egress filters. The IRRToolset, BGPQ3, and IRRPT are examples of open source tools that can produce router configuration for filters based on IRR data. [11]

2.2.2 RPKI

Many Internet incidents are caused by the spreading of false routing information, caused by the inability to verify if the ASs that are propagating the announcements are really authorized to do that. Indeed, information on Autonomous Systems and their prefixes may be found in public IRR databases that are not included in the BGP protocol. It does not naturally support establishing if an AS that announces a certain prefix on the Internet is authorized to do that, so an additional mechanism is required.

The Resource Public Key Infrastructure (RPKI) was created in February 2012 to address the issue that the information provided in IRRs is frequently missing or erroneous. It is a security framework that helps network operators make more informed and secure routing decisions. It is based on a public structure with distributed databases (RPKI repositories) including the associations between prefixes and its own authorised AS. Each pair of prefix and ASN is combined with a Digital Certificate, which allows users accessing the databases to validate the combination. This elements put together are called Route Origin Authorisation (ROA). The main purpose of digital certificates, cryptographic elements generated by security algorithms, is to validate public keys and the legitimacy of an AS to inject a given prefix block into the BGP using a given AS number.

A ROA contains 3 mains information:

- ASN authorised
- prefix owned by the authorised AS
- maximum lenght of the prefix

The figure below shows the structure of a ROA element.

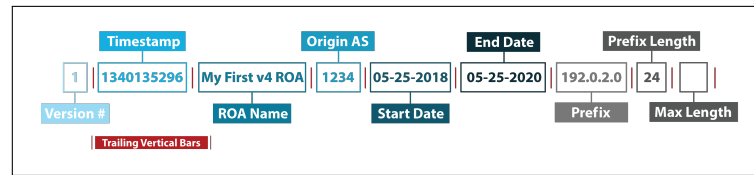


Figure 2.4. ROA structure [10].

Maximum Length attribute defines the maximum length of the IP prefix that the AS can announce. If it is not set, the AS is only permitted to announce the provided prefix. This method helps preventing hijacking by announcing a more specific prefix. [34]

The RPKI framework is based on repositories where ROA information is stored, which can be published via a particular Publication Protocol. On the basis of this structure, an AS can verify the correctness of received announcements and selectively choose those from certified sources. It is necessary for an AS to have trust that the information in the RPKI repository is correct and up to date on a global scale in order for the framework to work efficiently.

The figure below shows the structure of the RPKI framework with all its components.

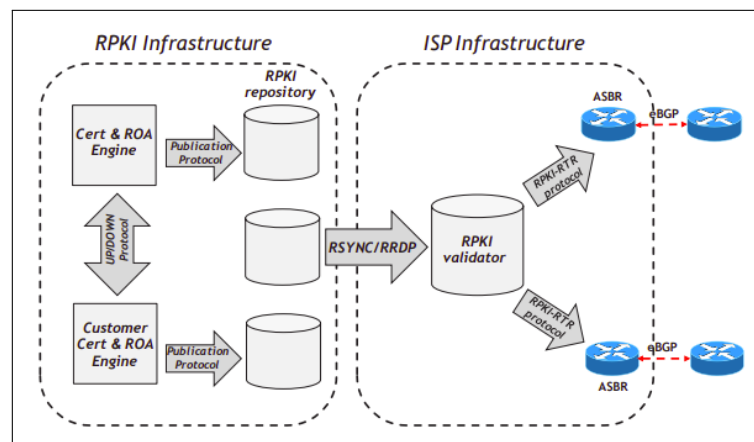


Figure 2.5. RPKI structure.

The architecture requires the use of servers called RPKI Validators, which themselves interface with RPKI repositories to perform a local download of ROAs. The ISP's Edge routers (ASBRs) locally download the ROAs present in the RPKI Validator and store them in their own RPKI Table. When a BGP announcement arrives, a router can thus verify the authenticity of the announcements it receives by comparing their content with that of the ROAs in its RPKI Table. The results of the prefix validation process can be: *Valid*, *Invalid* and *NotFound*.

The main benefits of using the RPKI lie in the certainty that routing information corresponds to certified resources, allowing prefix owners to prove the holdship of resources by distributing them to their customers. By using the certificate, the system is cryptographically protected and robust against attacks. Only prefix owners can then generate the correct association with their ASN. [35]

2.2.3 BGPsec

BGPSEC (Border Gateway Protocol Security) is a Border Gateway Protocol (BGP) extension that improves the security of BGP routing. RPKI is a first step in certifying BGP routing data, however it provides little or no security against a clever attacker if used alone. [20]

For example, there is no mechanism to prevent AS PATH manipulation. BGPsec aims to prevent the problem of path poisoning and it introduces a system of signatures to prove that the path has not been invented and basically it works based on encrypting the entire path.

BGPsec emerges as a BGP extension capable of introducing a level of security for this critical attribute. This protocol is however based on the RPKI infrastructure, which is required for validating that a certain ASN and a certain prefix have been assigned to the authorised operator. It extends RPKI introducing a new form of certificate known as a BGPSEC router certificate, which binds an ASN to a public signature verification key while the associated private key is owned by one or more BGP speakers inside this AS. In opposition to the RPKI infrastructure, BGPsec is software integrated into the BGP, which means that all the heavy validation-related operations will have to be performed by the router itself.

When two BGP routers agree to utilize BGPsec, the AS PATH attribute in BGP updates is replaced with a new BGPsec Path attribute, which provides the same feature in a secure way. When a regular BGP router originates a prefix, it generates an AS PATH attribute containing only the current AS number.

When a router sends a BGP update to a BGP neighbor in another AS, it attaches the local AS number to the left of the AS PATH. In BGPsec, BGPsec Path is produced instead of the AS PATH, they have the same functionality except that the router now includes the ASN to which the update is being sent, and then produces a cryptographic signature over the information added to the AS path. With each hop in the AS path now signed, a router receiving a BGP update with the BGPsec Path attribute may verify that the AS path is valid. [21]

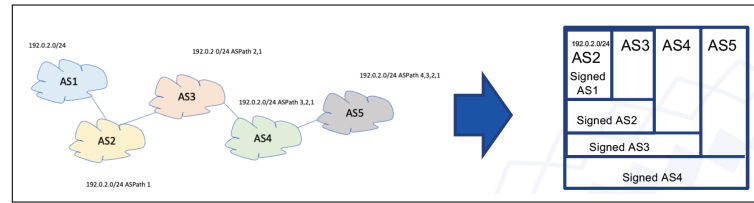


Figure 2.6. BGPsec signing chain.

BGPsec's security solution is effective, but it is too expensive in terms of computation and resources to be used on a large scale. Cryptographic verification can take a long time and ignores the fact that the future of BGP will see larger and more numerous messages, which will be difficult to manage with this security protocol. For these reasons, this security measurement is rarely used today and would require significant software/hardware improvements to function properly.

2.2.4 ASPA

BGPsec was created to address the issue of AS PATH integrity. However, the authors of a new Internet Draft, "Verification of AS PATH Using the Resource Certificate Public Key Infrastructure and Autonomous System Provider Authorization" [7], claim that, despite its complexity, its backward compatibility with insecure BGP allows an attacker to create an attack nullifying all AS PATH signing work. The authors propose a more realistic strategy to leverage the advantages of RPKI without the necessity for massive implementation of BGPsec. The new idea that propose to deal with Path Validation is based on the fact that any AS should declare its upstream providers and peers. Thanks to this declaration, the networks that can propagate its prefix announcements are known. The more networks that perform this, the more opportunities there are to discover misconfigurations. [8]

By definition:

ASPA is digitally signed object that bind, for a selected AFI, a Set of Provider AS numbers to a Customer AS number (in terms of BGP announcements not business), and are signed by the holder of the Customer AS. An ASPA attests that a Customer AS holder (CAS) has authorized Set of Provider ASes (SPAS) to propagate the Customer's IPv4/IPv6 announcements onward, e.g. to the Provider's upstream providers or peers. [6]

Even if it seems to be a good solution to Path Validation, route leaks with correct AS relationships cannot be protected by ASPA.

However, the relative simplicity of ASPA compared to BGPsec may make it more palatable to router vendors and network operators.

At the moment it's still under development and only its draft is available.

2.3 Recent BGP Incidents

In the following section, I will discuss the current situation of BGP security and show the results of a study I conducted in order to collect them in a single document. Finally, I will describe one of the most famous historical attacks that have affected the protocol, explaining the events and causes.

2.3.1 Security Bulletins and Statistics

BGP offers the flexibility and scalability required to support Internet expansion with minimal disruption. However, because it is reliant on trust and lacks built-in security mechanisms, issues do arise. When a network engineer makes a little error, or a bad actor redirects traffic on purpose, the result is inaccurate routing information that BGP accepts and distributes throughout the Internet. Examples of these bad events include route leaks, BGP hijacking, and IP address spoofing, all of which have the ability to slow down Internet connections or cause portions of the Internet inaccessible. Traffic can also be routed through malicious networks, permitting illegal inspection.

The past few years have disrupted certain mechanisms and stability of the Internet ecosystem. The technological infrastructure has had to cope with the enormous increase in usage caused by the COVID-19 pandemic. This event certainly attracted hacker attacks from all over the world, so it became very important to monitor and work on network security.

Every year, MANRS (Mutually Agreed Norms for Routing Security) analyzes incidents that highlight the vulnerability of the entire Internet routing ecosystem and how the vast majority of these events may be prevented by adhering to the best practices outlined in MANRS' programs. Using data from Cisco's BGPStream.com, one of the MANRS Observatory's data sources that offers information regarding suspicious routing occurrences. [3]

BGPStream collectors identified 2,477 BGP hijack occurrences in 2020, as shown in the graph below:

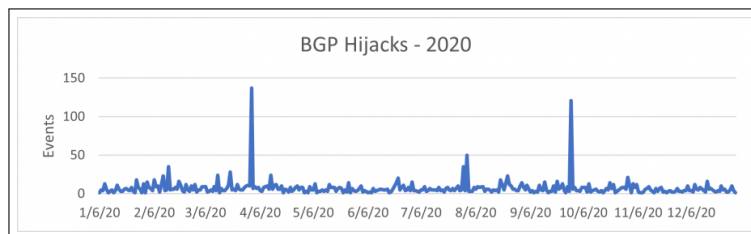


Figure 2.7. BGP Hijack 2020 bulletin.

In 2020, 1,396 BGP leaks incidents were detected, as shown in the figure below:

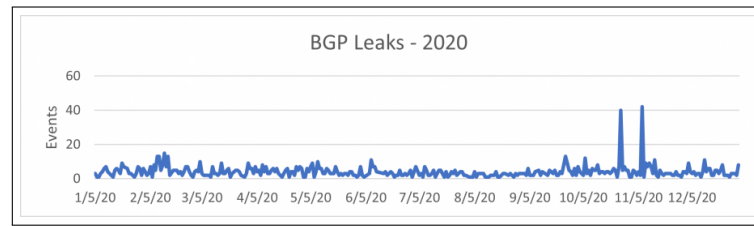


Figure 2.8. BGP Route Leaks 2020 bulletin.

The data were compared to those from 2019, and there was a significant increase in route leaks and a decrease in BGP Hijack.

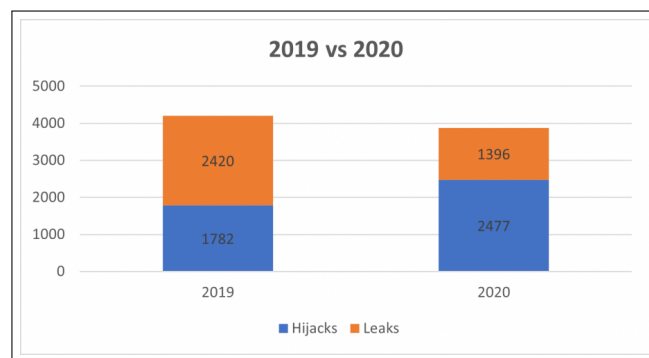


Figure 2.9. Differences between 2019 and 2020 Security Bulletin.

Many network operators took routing security seriously in 2020 and began taking steps to safeguard the global routing table. In particular, an increasing number of operators proactively implemented route filtering, established Route Object Authorization (ROAs), and began performing Route Origin Validation (ROV). [2] In 2021, the frequency of detected attacks dropped significantly. BGPStream collectors detected around 775 events classified as BGP Hijack.

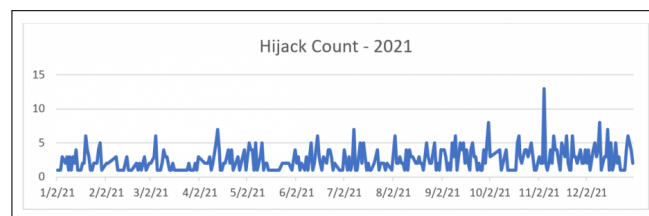


Figure 2.10. BGP Hijack in 2021 bulletin.

The good news is that there were fewer occurrences in 2021 than in 2020. However, 775 is still far too many, and there were a few events that caused huge delays throughout the world.

BGPStream collectors detected around 830 events classified as route leaks.

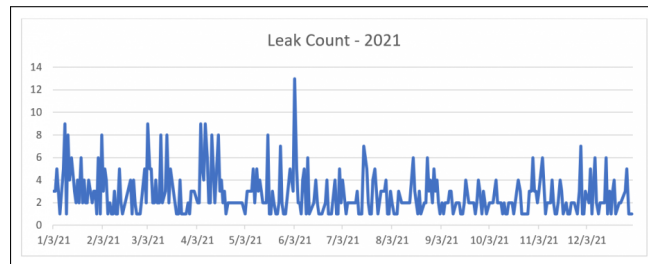


Figure 2.11. BGP Route Leaks in 2021 bulletin.

The marked decrease in attacks detected during 2021 demonstrates the strong effort of network operators to acquire appropriate security mechanisms for their protection. The global average RPKI infrastructure usage rate rose to around 34%, which is a good result but still far from optimal. [3]

Network technology communities such as MANRS have been doing activism to encourage network operators to operate securely, making the Internet a safer system for everybody.

We can summarise the last 3 years incident activity in the following schema:

	2019	2020	2021
BGP HIJACKS	1782	2477	775
BGP LEAKS	2420	1396	830

Figure 2.12. Last 3 years security bulletin.

2.3.2 Major Incidents Taxonomy

For educational purposes, I performed a research on some of the most recent and important attacks detected by multiple monitoring sources.

Route Leaks detected by Qrator

- On December 30, 2017, Idea Cellular Limited (AS55644) caused a major BGP route leak between its peers and upstream providers, including TATA (AS6453), Reliance Globalcom Limited (AS15412), and Sify (AS9583). This anomaly affected a massive number of networks all around the world, including content providers, transit ISPs, and more than 70000 prefixes in total. [30]
- AS39523 (DV-LINK-AS), a new interdomain routing ecosystem actor, began announcing its address space (one prefix), while also hijacking 80 high-profile prefixes. The prefixes that were hijacked belonged to both Russian and international content providers, including Google, Facebook, Microsoft, Mail.ru, V Kontakte, and many more. The incident lasted more than two hours, beginning at 4:44 UTC with 80 hijacked prefixes and ended at 7:19 UTC with another peak at 7:04 UTC. [29]
- On September 27, 2017, at 13:28 UTC, AS9299, which belongs to the Philippines' largest ISP - Philippine Long Distance Telephone Company (PLDT), announced prefixes between multiple Tier-1 operators (TATA, Cogent, Telecom Italia, PCCW) and AS1273, which belongs to Vodafone Europe. As a result, traffic from over 2000 prefixes in the United States, India, and the Philippines was routed to the Asia area. This anomaly's active phase lasted more than 2 hours. [31]
- At 15:03 UTC on September 5, 2017, AS55410, held by Vodafone India, leaked over 10,000 prefixes in the direction of AS1273, owned by the main Vodafone company located in Newbury, United Kingdom. This leak expanded to the rest of the world, including most Tier-1 ISPs. Three major Indian ISPs were directly affected (ASNs: 4755, 18101, and 9498), with increased latency in their networks. More than 400 operators in South Asia were indirectly impacted. This incident's active phase lasted 5 minutes, with a total leak length of 25 minutes. [32]
- On Tuesday, September 29, 2020, AS1221 - Telstra announced 472 prefixes in a BGP hijack incident that affected 266 other ASNs in 50 countries, with the biggest damage done to networks headquartered in the United States and the United Kingdom. It impacted over 1680 IPv4 prefixes worldwide, resulting in about 2000 route challenge conflicts. This issue had a three-hour peak in impacted prefixes, beginning at 17.50 UTC and declining below 400 affected prefixes until around 21.00 UTC. ROAs were in place for some of the hijacked prefixes, ranging from 20-25%. [27]

- On August 24, 2020, Qrator.Radar BGP monitoring detected a huge route leak originating from the AS42910 - Premier DC, having 1403 prefixes, the majority of which were from the United States (571) and, curiously, Akamai. Then there are nearly all of the nations in Western Asia. The incident began at 11:21 UTC and lasted four separate waves before terminating about 16 UTC. We can report that this particular routing anomaly lasted more than 4 hours in total, recurring numerous times with success. [28]
- A Brazilian AS264462 belonging to "Comercial Conecte Sem Fio Ltda me," as indicated in the whois record for this specific ASN, leaked significant 13046 network prefixes in a networking incident that lasted 1 hour and 23 minutes, beginning at 9.15 UTC and ended at 10.38. This route leak affected about 1995 ASNs originating in 104 countries. However, it drew more networks into this routing warp from Russia (345 ISPs/1236 prefixes) and South Korea (217 ISPs/1085 prefixes) than from anyplace else, with Germany taking third position with 136 ISPs and 478 prefixes. [26]
- On Sunday, April 5, 2020, an AS7552 belonging to Viettel - the largest telecommunications service provider in Vietnam, according to Wikipedia - was leaking routes for more than 3 hours in a row. The leak impacted 4825 network prefixes from 326 operators, spreading from AS7552 upstreams: AS3491 and AS4637 to AS1273 - Vodafone, which contributed in distributing it to nearly all major Tier-1 ISPs. The majority of networks in Vietnam, Cambodia, and Australia were disrupted, with more than 25% of ISPs in the first two nations affected. [33]

Route Leaks detected by Qrator and other sources

- AS12389 (Rostelecom) hijacked numerous prefixes from the main cloud networks on April 1, 2020. The ISP published prefixes belonging to Akamai, Amazon AWS, Cloudflare, Digital Ocean, and Hetzner, among other well-known companies, for nearly an hour. [14]
- MEZON hijacked 1029 prefixes on October 13, 2021, causing 1548 conflicts for 1152 prefixes and 251 ASNs in 16 countries. 79% maximum propagation. The incident last about 1 hour. [22]
- MEZON hijacked 3786 prefixes on October 25, 2021, causing 8324 conflicts for 4765 prefixes and 972 ASNs in 42 countries. 100% maximum propagation and lasted 36 minutes. MEZON-LT has hijacked worldwide BGP for the third time in a row, and it's just getting worse. [23]
- AS62325 — HDHK — hijacked 89 prefixes on September 21, 2021, causing 1354 conflicts for 1252 prefixes and 107 ASNs in 22 countries. 78% maximum propagation and lasted 14 minutes. The majority of announced prefixes are distributed via HK-IX (AS4635), with certain attributes circumventing HK-IX policies. [24]

- AS48467 — PRANET — hijacked 454 prefixes on May 18, 2021, causing 1011 conflicts for 166 ASNs in 40 countries. 75% maximum propagation and lasted 5 hours 41 minutes. Severity: high. [25]
- Pakistan Telecom (AS17557) began an unauthorized announcement of the prefix 208.65.153.0/24 on Sunday, February 24, 2008. One of Pakistan Telecom's upstream providers, PCCW Worldwide (AS3491), transmitted this announcement to the rest of the Internet, resulting in global YouTube traffic hijacking. [36]
- On 28 March, 2022, between 12:05 and 13:00 UTC it was detected a BGP hijack of the prefix 104.244.42.0/24, normally announced by AS13414 (Twitter). It was also announced by ASN 8342 (RTCOMM-AS, RU). The hijack didn't propagate far due to a RPKI ROA which asserted AS13414 was the rightful origin. [12]

From this list of incidents, it is instructive to pause and examine one of the historical BGP attack, which dates back over 15 years and is a good case study for understanding the chain of events that occurs during a BGP Hijack and how it is handled.

2.3.3 Example of Prefix Hijack

On Sunday, 24 February 2008, Pakistan Telecom (AS17557) started an unauthorised announcement of the prefix 208.65.153.0/24.

The incident was characterised by the following significant events:

1. AS17557 (Pakistan Telecom) begins announcing 208.65.153.0/24. The announcement is spread through AS3491 (PCCW Global). The announcement is received by routers all throughout the world, and YouTube traffic is routed to Pakistan.
2. AS36561 (YouTube) begins announcing 208.65.153.0/24. When two identical prefixes are present in the routing system, BGP policy rules, such as selecting the shortest AS path, decide which route is adopted. As a result, AS17557 (Pakistan Telecom) continues to get part of YouTube's traffic.
3. AS36561 (YouTube) begins announcing the prefixes 208.65.153.128/25 and 208.65.153.0/25. Every router that receives these announcements will route traffic to YouTube due to the longest prefix match policy.
4. All prefix announcements, are seen followed by another 17557 hop in their path. Because of the longer AS path, more routers choose the YouTube-originating announcement.
5. AS3491 (PCCW Global) removes all prefixes originating by AS17557 (Pakistan Telecom), thereby ending the hijacking of 208.65.153.0/24.

The incident is summarised in the following figure.

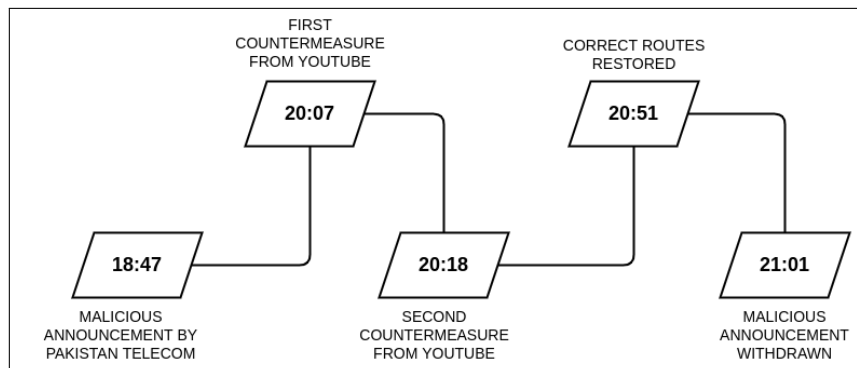


Figure 2.13. YouTube Prefix Hijack.

The hijacked prefixes were previously known thanks to reports on several mailing lists. Pakistan intended to limit access to YouTube. As the timeline above demonstrates, this incident occurred in a relatively short period of time: YouTube reacted roughly 80 minutes after the Pakistan Telecom announcements, and all significant events were completed within two hours. [36]

There's important to understand that unauthorised announcements like these can be stopped from propagating throughout the Internet by Autonomous System administrators using suitable routing configuration.

Chapter 3

Twitter Prefix Hijack

3.1 Case Study Description

On 28 March 2022, has been reported by several sources a BGP hijack of Twitter by a Russian ISP. From 12:05 to 12:50 UTC, Russian ISP RTComm (AS8342) hijacked a Twitter prefix (104.244.42.0/24). Fortunately, the hijack did not go far due to an RPKI ROA claiming AS13414 as the legitimate origin.

RTComm.ru is a significant Russian telecommunications company. Following Russia's invasion of Ukraine, which led to numerous Twitter posts disapproving of Russia's conflict, Russian ISPs began restricting access to Twitter. Hijacking a BGP prefix is one method of restricting access, but it may also be used to intercept traffic to specific IP addresses. In this case, it's unclear whether traffic interception is an objective. [15]

During this incident, RTComm (AS8342) exploited its upstream MTS PJSC (AS8359) to send the malicious announcement to as many networks as possible. MTS is an Autonomous System with almost 1,000 peers, as reported by bgp.he.net. [13]
















Rank	Description	IPv6	Peer
1	Level 3 Parent, LLC	 X	AS3356
2	Zayo Bandwidth	 X	AS6461
3	CLARO S.A.	 X	AS4230
4	Akamai Technologies, Inc.	 X	AS32787
5	Hong Kong Internet Exchange--Route Server 1	 X	AS4635
6	Filanco LLC	 X	AS29076
7	Telemar Norte Leste S.A.	 X	AS7738
8	PJSC "Vimpelcom"	 X	AS3216
9	Cox Communications Inc.	 X	AS22773
10	Singapore Telecommunications Ltd	 X	AS7473
11	Fortel Fortaleza Telecomunicacoes Ltda	 X	AS61832
12	COLT Technology Services Group Limited	 X	AS8220
13	Seabras 1 USA, LLC	 X	AS13786
14	InterNexa Global Network	 X	AS262589
15	GlobeNet Cabos Submarinos Colombia, S.A.S.	 X	AS52320

Figure 3.1. A portion of peers of AS8359.

The image above depicts a portion of MTS's peers, which includes several well-known networks such as Level 3 Parent and Zayo Bandwidth. The connection of MTS with other AS is significant for the effectiveness of the attack: the more numerous and vulnerable its peers are, the higher the probability of the attack spreading.

On February 5, 2021, a nearly identical incident occurred when an ISP in Myanmar attempted to hijack the exact same prefix. It's natural to wonder why this identical prefix was attacked again. The reason is most probably because Twitter's DNS A records point to this prefix, therefore blocking traffic to it might possibly affect traffic to Twitter.

The following are all of Twitter.com's DNS A records:

twitter.com.	776	IN	A	104.244.42.193
twitter.com.	776	IN	A	104.244.42.65
twitter.com.	763	IN	A	104.244.42.129
twitter.com.	1400	IN	A	104.244.42.1

Figure 3.2. DNS A records for twitter.com.

According to RIPE RIS data, AS8342 began originating 104.244.42.0/24 on Monday, March 28, 2022 at 12:06:11. (UTC).

1648469171.000000	104.244.42.0/24	199524	8359	8342
1648469171.000000	104.244.42.0/24	28917	8359	8342
1648469171.000000	104.244.42.0/24	31027	8359	8342
1648469171.000000	104.244.42.0/24	57695	60068	8359 8342
1648469171.000000	104.244.42.0/24	8359	8342	
1648469174.000000	104.244.42.0/24	199524	8359	8342
1648469174.000000	104.244.42.0/24	8660	1267	8359 8342
1648469176.000000	104.244.42.0/24	25091	8359	8342
1648469178.000000	104.244.42.0/24	56665	8359	8342
1648469184.000000	104.244.42.0/24	59414	13030	8359 8342
1648469184.000000	104.244.42.0/24	8359	8342	
1648469185.000000	104.244.42.0/24	21232	13030	8359 8342
1648469185.000000	104.244.42.0/24	25091	8359	8342
1648469191.000000	104.244.42.0/24	59605	8359	8342
1648469193.000000	104.244.42.0/24	41095	8359	8342
1648469195.000000	104.244.42.0/24	15435	8359	8342
1648469196.000000	104.244.42.0/24	8607	8359	8342
1648469199.000000	104.244.42.0/24	137409	8359	8342
1648469199.000000	104.244.42.0/24	25091	8359	8342
1648469199.000000	104.244.42.0/24	9304	8359	8342
1648469200.000000	104.244.42.0/24	22652	8359	8342
1648469201.000000	104.244.42.0/24	137409	8359	8342
1648469202.000000	104.244.42.0/24	19151	8359	8342
1648469207.000000	104.244.42.0/24	6720	8447	8359 8342
1648469207.000000	104.244.42.0/24	8660	1267	8359 8342
1648469210.000000	104.244.42.0/24	51185	8359	8342
1648471814.000000	104.244.42.0/24	31027	8359	8342
1648471824.000000	104.244.42.0/24	199524	8359	8342
1648471844.000000	104.244.42.0/24	19151	8359	8342

Figure 3.3. Malicious spread of Twitter's prefix.

AS8342 is connected to DataIX Internet Exchange, and its bgp upstreams include AS12389 (Rostelecom), AS8920 (RTComm), AS8359 (MTS), AS25091 (IP-Max), AS6939 (Hurricane Electric), and others.

The above figure illustrates that the propagation occurred exclusively through AS8359 (MTS), whereas others successfully filtered the erroneous/malicious BGP announcements. MTS (Mobile TeleSystems PJSC) is a well-connected network that spread this route over the worldwide routing table.

According to RIPE RIS statistics, this hijacked announcement was accepted by several networks via MTS and subsequently transmitted to their neighbors, however it did not travel globally and did not cause a global outage. The reason for this is Twitter's adoption of RPKI.

A Route Origin Authorisation (ROA) is a cryptographically signed object that specifies which AS is permitted to originate a certain IP address prefix or collection of prefixes. ROAs provide crucial protection against mis-origination. Many network operators have implemented Route Origin Validation (ROV), and any routes with a "Invalid" ROA status will be rejected. AS8342's network 104.244.42.0/24 was deemed invalid since its data did not match the Twitter-created ROA. Many operators rejected the route because they filtered it and stopped it from propagating further to the whole routing table.

It is critical for network operators to provide effective route filtering based on verified information about which networks are legally permitted to originate which quantity of resources (AS numbers and IP prefixes). This is why RPKI was created. [4]

3.2 Incident Analysis

I went on to the more in-depth analysis phase after acquiring enough information on the accident's course and causes. I examined this attack from many perspectives, supported by the use of different tools. The analytical procedure and results are described below.

3.2.1 Analysis by BGPlay Tool

Routing data is massive and, in its raw textual form, can be difficult to analyze and interpret. A better understanding may be obtained by graphically expressing the routing history of a collection of Internet number resources.

BGPlay visualizes changes in BGP routes related to an Internet number resource (IP prefix or origin AS) or a set of resources. It depicts the links between the BGP collection points and the target resource across all AS paths. It also animates this graph over time using ordinary 'play' controls. This allows you to visualize and comprehend a wide range of routing scenarios, including prefix announcements/withdrawals, multi-origin prefixes, single/multi-homing, route flapping, prefix hijacking, AS interconnectivity, and so on. RIS provides the BGP data used in the visualization employing its 13 Route Collectors (RRCs) throughout the world to collect data on BGP updates on a constant basis.

This data is filtered based on the user-specified selection of resources, time period, and collection points and is supplied in two computed sections through the RIPEstat data API:

- BGP initial state - at the start of the time period
 - showing all AS paths from Route Collectors to the group of prefixes/origin Autonomous Systems
- BGP updates - occurred in the complete time period
 - depicting the evolution of BGP routes throughout the time period specified

The graphical user interface of BGPlay, shown in the figure below, allowed me to manually analyze the progression of the attack step by step thanks to its simple controls.

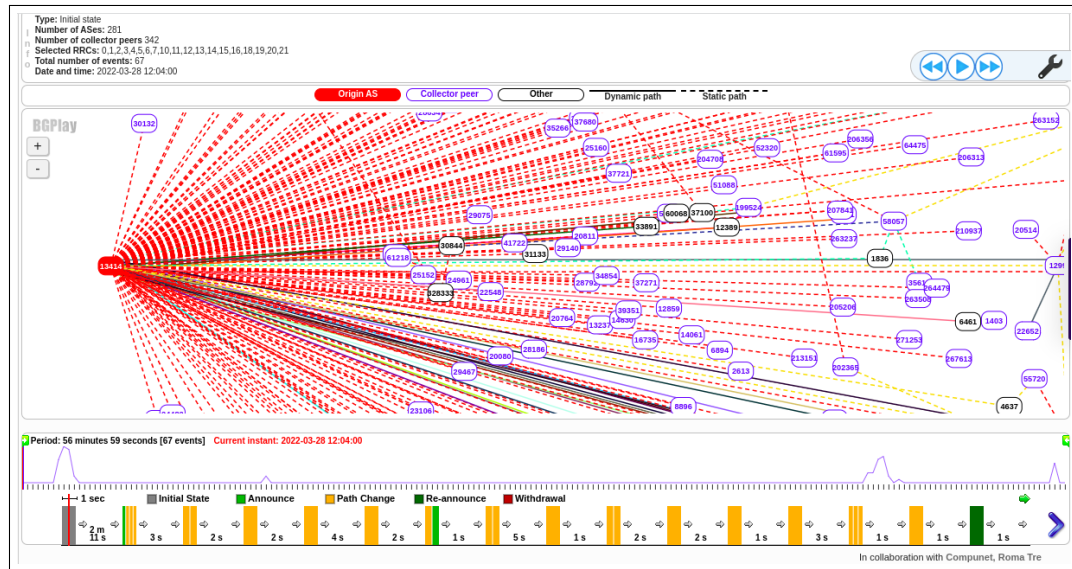


Figure 3.4. BGPlay GUI during Twitter Prefix Hijack.

The target origin AS is displayed in red; an AS peering directly with a route collector is shown in blue; and any other traversed AS is shown in black. When a BGP announcement occurs, a new path linking a group of nodes is animated. Each event that occurs along that path causes an animation that is particular to the event type. The timeline panel at the bottom is divided into two portions. The first, titled Control Timeline, depicts the number of events' trend over time. The second, called Selection Timeline, displays individual events in chronological sequence. [41] Analysis through BGPlay confirms that the attack began at 12:06 p.m., when AS8342 (RTComm) announced the Twitter prefix to its upstream, AS8359 (MTS). AS8359 propagates the announcement to several ASs, which elect 8359 8342 as the best-path to Twitter.

The first AS affected is 28917, and subsequently in the same way 57695, 199524, 8660, 25091, 25091, 56665, 59414, 21232, 59605, 41095, 15435, 8607, 137409, 9304, 22652, 19151, 6720, 51185 are also hijacked. At 12.50 p.m., the Twitter prefix announcement by 8342 was withdrawn and all ASs that had been hijacked returned to the correct path to the official Twitter AS13414.

In the following figures, the graphical reproduction of the hijacking visualised through BGPlay.

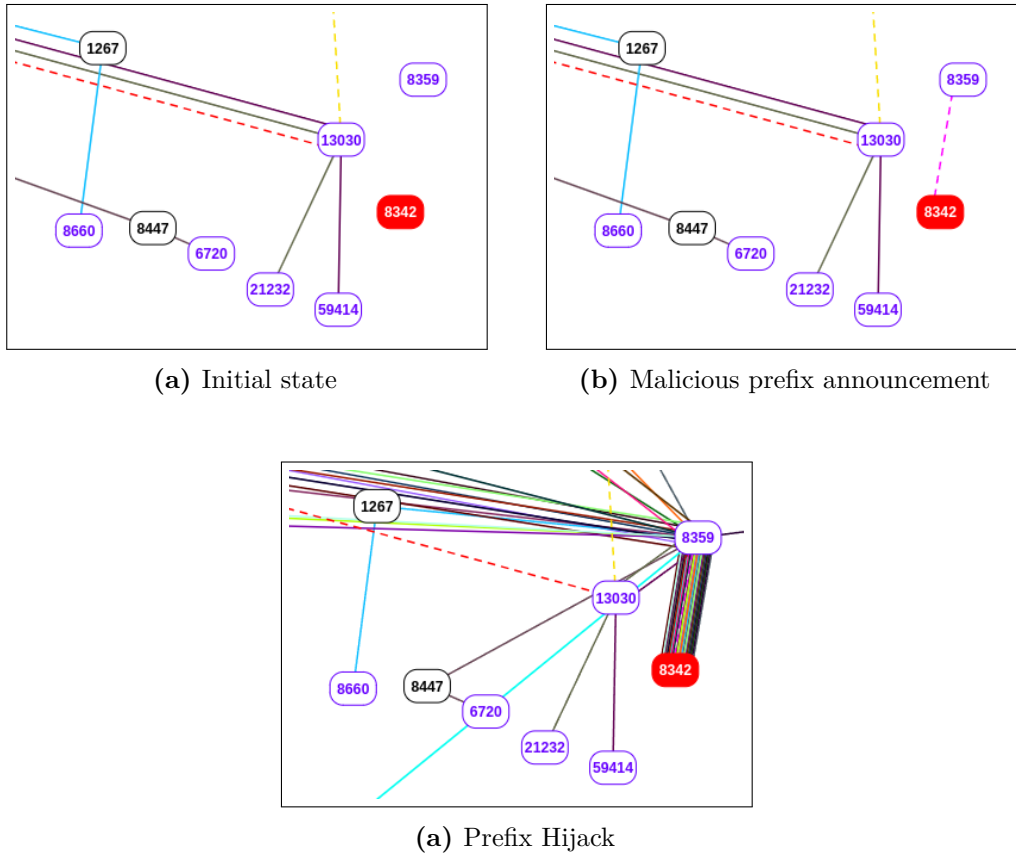


Figure 3.6. Progression of Twitter's prefix hijack with focus on AS8342.

The Twitter prefix had its corresponding ROA so the incident was contained, but with further care it would probably have had no consequences. The hijacking could have been avoided if upstream AS8359 had adopted origin verification measures such as RPKI. This manner, the malicious announcement that originated from RTComm would not have spread. Furthermore, all ASs that do not perform origin verification were most likely impacted by the prefix hijack.

3.2.2 ASs Involved and Relationship Analysis

After analyzing the hijack with BGPlay, I moved on to investigate the specifics of each Autonomous System involved in the attack individually in order to have a global vision of the context. To find the information, different tools were employed. First of all, I conducted a search based on Autonomous System DNS lookup. I used web tools like DNS Checker and bgp.he.net to do this. These tools make it possible to obtain all public details relating to an Autonomous System, like which company is using that ASN and what's the origin of that company, country, city, latitude, longitude, state, contact information as name, phone number and more. Moreover, WHOIS Lookup results reveal who or what entity owns or manages that domain name, name servers, IP address blocks. Some Autonomous Systems reveal important information about their bgp management, such as upstream, customers, peers, partners, and routing policies.

Another technique for studying Autonomous Systems is the use of bgp.he.net. This tool provides a variety of highly useful tools for defining an AS's identity and its relationships with other networks.

This online tool makes it possible to perform whois queries, visualise graphs representing the relationships between Autonomous Systems, consult their prefixes and peers, and find out the Internet eXchange points where they converge.

The results obtained during this research have been reported in the Appendix. A

For the purposes of the software development that will be explained in the following chapters, it was necessary to distinguish the relationships between Autonomous Systems in order to model the differences in their behaviour. To reach this objective, I conducted an in-depth study of the relationships that Autonomous Systems can have with each other through different web tools.

BGP is used to establish peering, which transmits routing information between two systems identified by their Autonomous System Numbers (ASNs). The BGP setting on both endpoints of the connection defines the type of the relationship.

There are two primary types of relationships that connect Autonomous Systems:

- **transit**
 - The networks are linked so that one (typically an ISP) can provide access to the whole Internet to the other, which is generally a "endpoint" entity (e.g., content or application provider, residential broadband provider). Most of the time, there is a business relationship. The endpoint entity pays the ISP for transmitting traffic to and from the Internet.
- **peering**
 - The networks exchange only traffic that originates or terminates within their respective networks (or the networks of their direct customers). It is an effective strategy for businesses companies that brings several benefits to both parties consisting of reducing the cost of their transit and the complexity of their connection, improving security and performance of the network. [17]

These relationships are often implemented through commercial agreements, which translate into engineering regulations on traffic flows inside and between various networks that participate in the global Internet routing system.

AS relationships govern routing policies, which impose a nontrivial set of limits on the paths through which Internet data can transit. This has repercussions for network robustness, traffic engineering, and macroscopic topology measurement strategies, as well as other research and operational considerations.

The type of relationship Autonomous Systems have with each other seems to be not publicly available; I've never found certified entity that collects information about their relationships. This is most likely due to commercial and privacy concerns; probably the Internet community does not believe it is essential to make this information public. In order to achieve the most accurate work attainable, I cross-referenced the results of various researches and developed scripts to speed up the process. Below I describe the methodology used to obtain the final result.

The tools employed to conduct the study were mainly the following:

- bgp.he.net
- CAIDA Dataset
- bgpview.io
- Route-Views Collectors and peval

As a first step, I worked on the CAIDA dataset. CAIDA provides datasets that contain AS relationship inferred from BGP using specific methods. They produce monthly datasets that can be freely consulted. For the purpose of my research, I selected the dataset dating back to April 2022, to display the records for the month in which the prefix hijack occurred. The format of the dataset distinguishes transit relationships from peering in the following way.

Transit relationship:

$$provider - as > | < customer - as > | - 1$$

Peering relationship:

$$< peer - as > | < peer - as > | 0 | < source >$$

I developed a short python script that took the Autonomous System Number (ASN) as input and returned the list of its peers, its customers and its providers.

```

1 with open(str(sys.argv[1])) as f:
2     for line in f:
3         if line[0] == '#':
4             continue
5         else:
6             split = line.split("|")
7             for asn in my_as_complete:
8                 if my_as in split and str(asn) in split:
9                     if split[-2] == "0":
10                        peers.add(asn)
11                     elif split[0] == str(asn):
12                        providers.add(asn)
13                     elif split[1] == str(asn):
14                        customers.add(asn)

```

Listing 3.1. Python script that elaborates CAIDA dataset.

Below is an example execution of the script, in addition I report the full results in the Appendix. B

```
List of peers: {25091, 41095, 8607, 60068, 8359, 137409, 15435, 59605, 9304,
56665, 57695, 33891, 199524, 12389, 13030, 13414, 51185, 28917, 8447}

List of customers: {8660}

List of providers: {1299, 3356}
```

Figure 3.7. Output of CAIDA script example execution.

This approach gave me a first picture of the relationships between Autonomous Systems, but since the results were not completely accurate, I verified them with some other different tools.

The second method I used for this study involves the combination of two tools: the peval tool and routeviews collectors.

Peval is a tool that is used from the terminal and takes an ASN as input to return its prefixes. An example of peval output is shown in the figure below.

```
sara@bruzzese:~$ peval as8660
({213.209.0.0/18, 213.209.44.0/24, 213.209.40.0/24, 213.209.36.0/24, 213.209.32.0/24,
213.209.33.0/24, 213.209.34.0/24, 213.209.35.0/24, 213.209.30.0/24, 213.209.31.0/24,
213.209.27.0/24, 213.209.19.0/24, 213.209.16.0/23, 213.209.16.0/24, 213.209.17.0/24,
213.209.0.0/20, 213.209.7.0/24, 213.209.0.0/23, 213.209.0.0/24, 213.209.1.0/24, 212.48.0.0/19,
212.48.22.0/23, 80.93.64.0/20, 80.93.64.0/21, 80.93.72.0/21, 80.93.64.0/22, 80.93.68.0/22,
80.93.72.0/22, 80.93.76.0/22, 80.93.79.0/24, 80.93.66.0/24, 80.93.67.0/24})
```

Figure 3.8. Output of peval example execution.

Once the prefixes are obtained via peval, they can be used to make investigative calls on routeviews collectors via the terminal. The result is a BGP Routing Table that records the best as-paths to reach that Autonomous System, as shown in the below figure.

```
BGP routing table entry for 217.107.208.0/20
Paths: (33 available, best #31, table default)
Not advertised to any peer
34224 3356 12389 8342
 94.156.252.18 from 94.156.252.18 (94.156.252.25)
  Origin IGP, metric 0, valid, external
  Community: 34224:333
  Last update: Sun Jun 26 10:35:28 2022
3130 1239 3356 12389 8342
 147.28.7.2 from 147.28.7.2 (147.28.7.2)
  Origin IGP, metric 0, valid, external
  Community: 1239:321 1239:1000 1239:1010
  Last update: Wed Jun 22 10:52:02 2022
3303 12389 8342
 217.192.89.50 from 217.192.89.50 (138.187.128.158)
  Origin IGP, valid, external
  Community: 3303:1004 3303:1006 3303:1030 3303:3056 8342:42000 8342:47777
  Last update: Wed Jun 22 10:39:14 2022
2497 12389 8342
 202.232.0.3 from 202.232.0.3 (58.138.96.255)
  Origin IGP, valid, external
  Last update: Thu Jun 23 10:18:30 2022
```

Figure 3.9. Output of routeviews collectors example call.

From this file, as-paths can be analysed to determine the Autonomous System's upstreams. This approach does not have a high degree of accuracy, but when paired with the others, it can confirm and support previously discovered information. To rapidly analyze the documents, I wrote a Python script that returns a list of probable upstreams of an Autonomous System.

```

1 with open('as'+my_as+'.txt') as f:
2     for line in f:
3         splitted = line.split()
4         if splitted[0].isnumeric() and len(splitted) >= 2 and my_as
in splitted:
5             if my_as+', ' in splitted:
6                 index_as = splitted.index(my_as+', ')
7             else:
8                 index_as = splitted.index(my_as)
9             upstream = int(splitted[index_as-1])
10            if upstream in my_as_complete:
11                upstreams.add(upstream)
12            if upstream not in upstreams_dict:
13                upstreams_dict[upstream] = 1
14            else:
15                upstreams_dict[upstream] += 1

```

Listing 3.2. Python script that elaborates routeviews collector output.

This script outputs the ASNs directly adjacent to the one under examination, implying that they are its upstreams. For example, in the figure 3.9, AS8342 is always preceded by AS12389, suggesting that this is its upstream. To support this hypothesis, my script counts the amount of times the upstream appears. If this number is more than one or two, the probability increase and we can assume that it is upstream. The full results of this study can be found in the Appendix. C

The last two methods employed are the web tools bgp.he.net and bgpview.io, two excellent tools that analyse the network and report information on it. Both tools have an upstream section, which I show in the figure below.





IPv4 Upstreams		IPv6 Upstreams	
Country	ASN	Name	Description
	AS3356	LEVEL3	Level 3 Parent, LLC
	AS1299		Arelion, f/k/a Telia Carrier
	AS6461	ZAYO-6461	Zayo Bandwidth
	AS1239	SPRINTLINK	Sprint

Figure 3.10. Upstreams of AS1267 provided by bgpview.io.

The combination of all these techniques allowed me to establish a reasonably precise profile of the relationships between Autonomous Systems at the moment of the incident. This information was crucial for the construction of the virtual laboratory in which to carry out the experiments that we will see in the following chapters.

3.3 Selection of AS-subnetwork

After defining the relationships among the Autonomous Systems involved in the incident, I selected those to reproduce in the experimental laboratory on Kathará. The choice to make the selection was based on two main factors: the first was that reproducing a piece of the network in the laboratory would be the same as reproducing the entire network, and the second was the quantity of data that would need to be analyzed in order to totally emulate it. Then, I selected the Autonomous Systems and links that seemed to be the most significant and produced a graph to graphically illustrate their connections and relationships.

The Autonomous Systems selected for this purpose were AS8342, AS8359, AS1267, AS8660, AS21232, AS59414, AS1299, AS22652, AS13030 and AS13414.

Below is the graph I created with the descriptive legend.

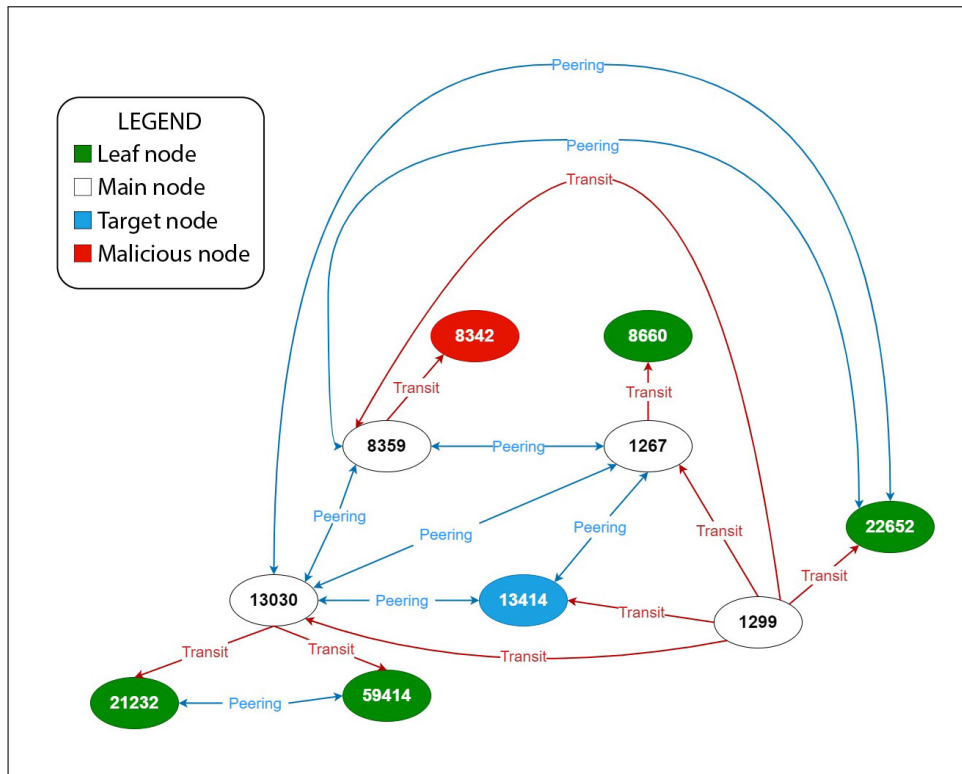


Figure 3.11. Graph of selected Autonomous Systems.

Starting from this graph, I was able to begin the software development work for the realisation of the experimental laboratories on Kathará.

Chapter 4

Kathará Environment

4.1 Origins and Scope

Before discussing what Kathará is and how it works, I believe it is necessary to clarify why it was created and what problems it seeks to fix.

Computer networks could be very complex due to the large amount of technologies involved. The network involves several devices (computers, smartphones, routers, etc.), interfaces, protocols, physical interconnections that lead to complex topologies.

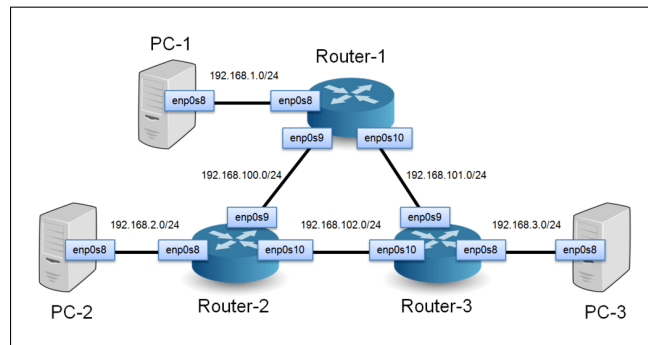


Figure 4.1. Example of a network topology.

Performing experiments on such complex networks may be unfeasible. A live network cannot be used for experimentation since it contains services that are vital to the organization and it would be necessary to coordinate multiple divisions of the company.

Moreover, to do this, you need to provide a specialised team and equipment. Network equipment is costly, and even for basic studies, numerous pieces of equipment should be present on the same test bed.

Another aspect that it's important to emphasise is the difference between simulation and emulation. Emulation and simulation systems offer users a virtual environment that may be used for testing, experiments, and measurements.

The goal of simulation systems is to replicate the performance of a real-world system (latency time, packet loss, etc.)

Emulation systems are designed to accurately reproduce the functionalities of a real-world system (configurations, architectures, and protocols), with little regard

for performance.

These concerns inspired the idea of developing a tool that would allow users to emulate networks and conduct tests on them.

Kathará is a system that emulates computer networks using Docker container and in the next section I'll explain how it works.

4.2 Characteristics and Functioning

Kathará is an open source container-based network emulation system for showing interactive demos/lessons, testing production networks in a sandbox environment, or developing new network protocols. [38]

Kathará helps to simplify the development of complex network scenarios.

It simulates network scenarios by employing a backend virtualization system such as Docker or Kubernetes. It requires low resource consumption to achieve quick deployments on a laptop using Docker. For this purpose, it provides several ready-to-use images like Base, Quagga, FRRouting, etc.

First of all, a container is a standard unit of software that wraps up code and all its dependencies so that the program can be migrated from one computing environment to another quickly and reliably.

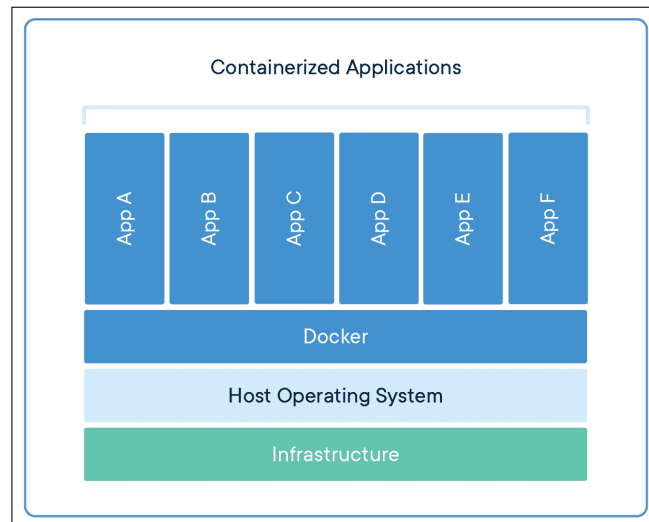


Figure 4.2. Docker container.

The basic idea of Kathará is that several containers are generated within a single host system, and the containers are connected to virtual collision domains and can thus communicate between themselves. Each container can be defined as a device that acts as a conventional host, a router, a switch, etc. Containers that are interconnected by virtual Layer-2 LANs emulate network devices. Each device has its own console, memory, filesystem and network interfaces.

Each network interface is associated with a single (virtual) collision domain, and each virtual collision domain can be associated with many network interfaces.

Kathará is a Python implementation of the best known Netkit. The framework has the performance to run in production, being ten times faster than Netkit and more than 100 times lighter.

To summarise: each network device in the Kathará environment is implemented by a container, and each interconnection link is emulated by a virtual network.

In this section, we will briefly look at how Kathará works in practice, some of its commands, settings and how it can be used.

Kathará is a tool available for Windows, MacOS and Linux systems.

It offers three sets of commands to users:

- v-commands
- l-commands
- global commands

V-commands are low-level utilities used to configure and run a single device with arbitrary configurations.

- **vstart** - starts a new device
- **vconfig** - attaches network interfaces to a running device
- **vclean** - halts a device

L-commands offer an easier-to-use platform for configuring extensive laboratories with several devices.

- **lstart** - starts a Kathará laboratory
- **lclean** - halts all the devices of a laboratory
- **lrestart** - halts all the devices of a laboratory and start them again
- **linfo** - provides information about a laboratory

Global commands are mainly management commands.

- **check** - check your system environment
- **connect** - connect to a running Kathará machine
- **list** - show all running Kathará machines of the current user
- **settings** - show and edit Kathará settings
- **wipe** - delete all Kathará machines and links, optionally also delete settings

A Kathará lab is a collection of preset devices that may be launched and stopped together.

A simple Kathará lab is a directory structure that contains the following files:

- **lab.conf** - file describing the network topology
- a series of **subdirectories** containing each device's setup settings
- **startup** files that explain the activities taken by devices when they are started

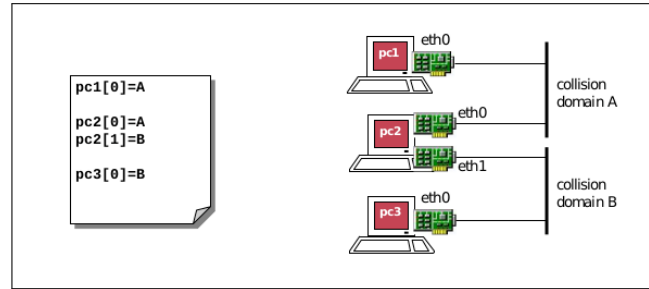


Figure 4.3. Example of lab.conf.

In Kathará, there are two methods for sharing files across the host and device filesystems:

- share files *mirrored* to the device
 - change inside the device will reflect in the host filesystem and viceversa
- share files *copied* to the device
 - two independent copies of the same files

4.3 Application

In the previous paragraphs, we briefly discussed why the Kathará environment was created, about its functions and capabilities. To conclude, I will discuss its importance in my research and why it was employed in my experiments.

My purpose was to use a framework to reconstruct networks, devices, and communications in order to conduct test and experiment on them. As we have seen, Kathará is an open-source system with several functionalities that are suitable for my necessities, as well as manuals, tutorials, and laboratories that are ready to be examined and used.

Its simplicity of use plays a crucial role in the design of complex networks; only a few files and settings are required to emulate the network in question.

Kathará helped me to analyze how the network works, establish an unlimited number of networks, investigate how protocols work and conduct security testing without having to rely on sophisticated software or setups that would have burdened my study.

I created the three core labs of my research based on Kathará, which aim to demonstrate the utility and necessity of BGP protocol countermeasure systems.

I used the libraries of Kathará to simulate the Autonomous System with its prefixes, to connect them together with virtual links and make them communicate.

Thanks to Kathará, I had complete control over every host in the emulated system, allowing me to manage the machines and conduct in-depth analysis in every moment.

In fact, Kathará will provide a root shell for each host in the network, from which commands can be executed. In the next chapters, I will explain in detail the implementation and result of my experiments on Kathará.

Chapter 5

KathBuilder Software Development

The analysis of the BGP protocol, its vulnerabilities, and related attacks has led to considerations about the utility of having a tool to reproduce incidents and run testing in a secure and constrained environment. Kathará, as described in the previous chapter, is a framework with good emulation and testing capabilities for basic and complex networks, allowing the use of various protocols and security measures. The idea of developing a testing environment to reproduce past incidents arose from a collaboration between the Internet Exchange Point (IXP) NameX in Rome and the University of Rome La Sapienza.

Within the industry environment, a tool that would allow the emulation and analysis of attacks with the possibility of being able to manipulate the environment at will during simulations, was evaluated as enriching. This software was therefore created to meet this need and to demonstrate how, by applying correct countermeasures to past incidents, it would be possible to avoid or contain them. The effectiveness of modern security mechanisms is proven, but unfortunately still few rely on them. There are many reasons for this happening, the main factors are the complexity of implementation and lack of awareness of the problem.

This software was developed in python language entirely by me, the work was divided into phases and developed through *agile* working methodology.

The purpose of this software is to create, in the most automated way possible, the structure of the laboratories that Kathará uses for emulations. In the following paragraphs, I will explain how the software works, the inputs required in order to properly work, and I will show an example of output.

5.1 Input

Before being able to run the KathBuilder software, it is necessary to carry out an upstream study and produce files that will be required for the automatic configuration of the network laboratory.

The input files used by the software consist of the two following:

- *data.json*
- *relationships.json*

Two json files were utilized since the program can easily process them thanks to the python packages that facilitate their usage.

The *data.json* file was provided directly by the BGPlay tool. In fact, this web tool not only allows the network status to be graphically reproduced, but it also makes data available for the given time interval. BGPlay takes as input several parameters such as the target prefix to monitor, the start time and the end time for the query and the list of Route Collectors to get the results from.

It gave in output the json file that comprehends:

- *initial_state* - the state of the BGP routes for the target prefix at the start time
- *events* - the BGP updates observed for the target prefix during the query time interval
- *nodes* - descriptive information of all the ASes present in the AS paths of the BGP routes/updates that comprehend the ASN and the owner organization
- *targets* - list of all the target prefixes present in the results
- *sources* - descriptive information of all the RIS collectors peers through which the BGP routes/updates were observed
- *query_starttime* - defines the start of the time interval covered in the query
- *query_endtime* - defines the end of the time interval covered in the query
- *resource* - defines the resource used in the query

The specifications of the json file can be found in the BGPlay documentation. [37] The *relationships.json* file, on the other hand, must be created manually, as a result of a study conducted on the incident to be emulated. To create this file, I used the graph in the figure 3.11, after analysing the Twitter prefix hijack incident.

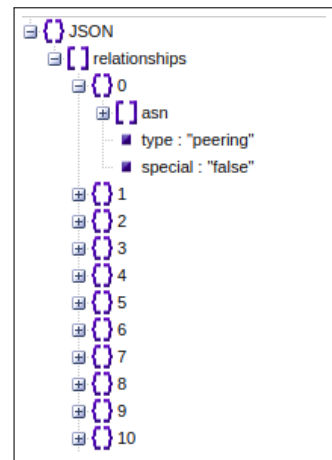
The purpose of this file is to express the graphical information contained in the graph through a handy json file.

The file consists in a list of elements, each identifying and describing the relationship between two Autonomous Systems. The attribute 'special' is used to identify the relationships between main AS, which will be treated differently during configuration phase.

The figures below show the structures of the two json files respectively.



(a) Json structure of *data.json*



(b) Json structure of *relationships.json*

Thanks to these two files, it's possible to obtain the whole context of the incident in question and enrich it with more precise and essential information, such as the type of relationship between Autonomous Systems. With these inputs, the KathBuilder software may be run to generate a Kathará laboratory that as closely as possible reflects the reality of the network.

5.2 Processing

The entire software consists of three modules, one with the main body and two performing external data processing operations.

The main body consists mainly of the following functions:

- identification of network connections
- obtaining network prefixes
- obtaining relationships between ASs
- creation of directories and subdirectories
- writing network configuration files of laboratory devices

First of all, by **identification of network connections**, means the processing of the data.json file obtained as input and the correct production of all connections between the Autonomous Systems of the targeted incident.

The function below shows one of the processing steps of the json, from which the paths are extrapolated in their initial state, before the incident occurred.


```

1 def create_initial_paths(paths):
2     initial_paths = []
3     for state in data["data"]["initial_state"]:
4         path = state["path"]
5         path_set = set(path)
6         for bad_path in paths:
7             i = bad_path.index(BAD_UPSTREAM)
8             asn = set()
9             for index in range(0, i):
10                 asn.add(bad_path[index])
11                 if asn.issubset(path_set):
12                     initial_paths.append(path)
13 initial_paths = set(tuple(i) for i in initial_paths)
14 return initial_paths

```

Listing 5.1. Path elaboration function.

The function selects only the Autonomous Systems and paths impacted by the incident, thus filtering essential information for a contextualised reproduction of the network of interest. Subsequently, individual Autonomous Systems are extracted from the paths and a set containing them is created. This first step is necessary in order to obtain data on the Autonomous Systems to be generated in the laboratory and how they are interconnected.

The process of **obtaining network prefixes** is referred to one of the two external module. For simplicity of implementation, each Autonomous System in the laboratory owns two prefixes among all its real prefixes. They are chosen randomly through a tool called *peval*. This tool makes it possible to obtain the set of prefixes of a given Autonomous System. From the list of prefixes returned by *peval*, the software selects the first two prefixes and assigns them to the Autonomous System in question. Before being assigned, the module checks that the two prefixes do not overlap, in which case a new prefix is selected.

```

1 def peval(my_as_list):
2     cmd = "peval"
3     outputlist = []
4     for asn in my_as_list:
5         counter = 1
6         temp = subprocess.Popen([cmd, "as"+str(asn)],
7                                 stdout=subprocess.PIPE,
8                                 stderr=subprocess.STDOUT)
9         stdout, stderr = temp.communicate()
10        outputlist.append(str(stdout))
11        lista = outputlist[0].split(",")
12        for i in range(0, len(lista)):
13            lista[i] = lista[i].strip()
14        ip1 = ipaddr.IPNetwork(lista[counter])
15        counter += 1
16        ip2 = ipaddr.IPNetwork(lista[counter])
17        while ip1.overlaps(ip2):
18            counter += 1
19            ip2 = ipaddr.IPNetwork(lista[counter])
20        asn_prefixes[asn] = [str(ip1), str(ip2)]
21        outputlist = []
22    return asn_prefixes

```

Listing 5.2. Peval elaboration module.

The second external module, on the other hand, deals with **obtaining the relationships between ASs** reading the `relationships.json` file. The information is stored and will later be used to correctly produce router configuration files. This same module consists of other functions that provide manipulation of the json file. The **creation of directories and subdirectories** phase consists of several functions; for example, the one that creates the lab directory and recursively all its contents, the function that produces the `lab.conf` by allocating interface numbers and network domains to each Autonomous System.

This phase is a bit complex since a function is required for each file to be written and the manipulation of specific resources to be accessed.

The function below shows the creation of the `lab.conf` by assigning interfaces and network domains.

```

1 def create_interfaces_and_lab_conf(relationships,
2   interface_dictionary):
3     interfaces = set()
4     using_dictionary = interface_dictionary.copy()
5     lab_conf_path = os.path.join(get_lab_directory(), "lab.conf")
6     f = open(lab_conf_path, "w")
7     ch = 'A'
8     for relation in relationships:
9         f.write("as"+str(relation[0])+"["+str(using_dictionary[relation
10         [0]]-1)+"]="+str(ch)+"\n")
11         f.write("as"+str(relation[1])+"["+str(using_dictionary[relation
12         [1]]-1)+"]="+str(ch)+"\n")
13         interfaces.add((relation[0], relation[1], using_dictionary[
14         relation[0]]-1, using_dictionary[relation[1]]-1))
15         ch = chr(ord(ch) + 1)
16         using_dictionary[relation[0]] -= 1
17         using_dictionary[relation[1]] -= 1
18     f.close()
19     return interfaces

```

Listing 5.3. Lab.conf create function.

The final phase, **writing the network configuration files of the laboratory devices**, is both complex and delicate. As aforementioned in previous chapters, it takes very little to incur in misconfiguration issues, thus it is critical for the laboratory's proper operation that this portion of the software works properly.

In this last phase, router configuration files are written, IP addresses are assigned to all machines, and `bgpd` and `fr` network services are configured.

One of the most complex functions is the one that generates the `bgpd.conf` files, which in themselves are often very long and rich in information such as the details of the Autonomous System, of the connections with its neighbors, the filters applied to each of them, the use of security metrics, use of log services and much more.

The following page is dedicated to the most important function of the laboratory, which includes almost all the creation and writing of the laboratory files, from the startup files of each machine, to the configuration of the *quagga* network service, to the writing of the *bgpd.conf* and of the *daemons* files.

```

1 def create_configuration_files(complete_set, pair_interface_tuples,
    prefixes_dict, interface_dictionary):
2     for node in complete_set:
3         #Create the startup file path
4         startup_path = os.path.join(get_lab_directory(), "as"+str(node)+
            ".startup")
5         clean_startup_files(node)
6         #Create the /etc/quagga path
7         node_directory = os.path.join(get_lab_directory(), r"as"+str(node)
            +"/etc/quagga")
8         if not os.path.exists(node_directory):
9             os.makedirs(node_directory)
10        #Create and write the /etc/quagga/daemons file
11        write_daemons_file(node_directory)
12    for element in pair_interface_tuples:
13        startup_path_first = os.path.join(get_lab_directory(), "as"+str(
            element[0])+".startup")
14        interface_number_first = element[2]
15        ip_fourth_counter += 1
16        ip_address_first = "11.0."+str(ip_third_counter)+"."+str(
            ip_fourth_counter)
17        ip_fourth_counter += 1
18        startup_path_second = os.path.join(get_lab_directory(), "as"+str(
            element[1])+".startup")
19        interface_number_second = element[3]
20        ip_address_second = "11.0."+str(ip_third_counter)+"."+str(
            ip_fourth_counter)
21        ip_fourth_counter += 2
22        write_ip_address_30(startup_path_first, ip_address_first,
            interface_number_first)
23        tuple_configuration.add((element[0], interface_number_first,
            ip_address_first))
24        write_ip_address_30(startup_path_second, ip_address_second,
            interface_number_second)
25        tuple_configuration.add((element[1], interface_number_second,
            ip_address_second))
26    del prefixes_dict[TARGET_AS]
27    prefixes_dict[TARGET_AS] = [ip_target]
28    for asn in prefixes_dict:
29        startup_path = os.path.join(get_lab_directory(), "as"+str(asn)+
            ".startup")
30        prefixes_list1 = prefixes_dict[asn][0].split("/")
31        ip_1 = prefixes_list1[0][:-1]
32        ip_tot1 = ip_1+"1/"+prefixes_list1[1]
33        if asn == TARGET_AS:
34            write_ip_address(startup_path, ip_tot1, interface_dictionary[
                asn])
35        else:
36            prefixes_list2 = prefixes_dict[asn][1].split("/")
37            ip_2 = prefixes_list2[0][:-1]
38            ip_tot2 = ip_2+"1/"+prefixes_list2[1]
39            write_ip_address(startup_path, ip_tot1, interface_dictionary[
                asn])
40            write_ip_address(startup_path, ip_tot2, interface_dictionary[
                asn]+1)
41    write_zebra_start(complete_set)

```

Listing 5.4. Configuration files create function.

5.3 Output

The execution of the software produces the laboratory complete with all the files necessary to be started and used, fully functional. The structure of the laboratory and the related files will be explored in the next chapter. In addition to the creation of the laboratory, the software prints on the console a series of useful information for consulting the network, which we will see in detail. The results shown in this paragraph are for illustrative purposes only and do not fully reflect reality.

As a first output we see the content of the relationships.json file printed, reworked and exposed in a more readable way to the user.

```
Relationships:
{(1267, 8359, 'peering', 'true'),
 (1299, 13414, 'transit', 'false'),
 (1299, 22652, 'transit', 'false'),
 (8359, 13414, 'peering', 'false'),
 (22652, 13030, 'peering', 'false')}
```

Then, we can see the full list of the ASs involved in the laboratory.

```
ASs complete list:
{13414, 13030, 8359, 21232, 1267, 8660, 1299, 59414, 8342, 22652}
```

Next, a dictionary that contains the association between each Autonomous System and the number of its interfaces, that is, of its connections with other ASs.

```
Interface dictionary:
{1267: 5
 8660: 1,
 13030: 7
 22652: 3,
 59414: 2}
```

A very important piece of information is to know the association between communication interfaces between the ASs, necessary to write the lab.conf and to test the traffic in the lab.

```
Pair interfaces:
{(1267, 8359, 2, 5),
 (1267, 8660, 3, 0),
 (1299, 22652, 1, 1),
 (13030, 13414, 2, 2),
 (22652, 13030, 2, 6)}
```

Equally important is knowing the IP address associated with each interface of each machine in the laboratory. This information allows it to write startup files and bgp configuration files.

```

Tuple configuration:
{(1267, 0, '11.0.0.62'),
 (1299, 0, '11.0.0.5'),
 (13030, 6, '11.0.0.46'),
 (13414, 0, '11.0.0.9'),
 (22652, 2, '11.0.0.45'),
 (59414, 0, '11.0.0.34')}

```

One of the last information returned, reports the IP address of the network on the link that connects two Autonomous Systems. All the addresses were chosen / 30 to allow only two hosts in the network, that is, the two communicating routers.

```

Ip network tuples:
{(1267, 8359, '11.0.0.68/30'),
 (1299, 1267, '11.0.0.28/30'),
 (8359, 8342, '11.0.0.20/30'),
 (13030, 1267, '11.0.0.60/30'),
 (13414, 1267, '11.0.0.8/30'),
 (21232, 59414, '11.0.0.16/30'),
 (22652, 8359, '11.0.0.56/30')}

```

Finally, the dictionary of prefixes associated with the Autonomous Systems, obtained through peval, is shown. These addresses will be used in the bgpd.conf to be advertised on the network.

```

Prefixes dictionary:
{1267: ['212.245.0.0/16', '212.141.0.0/16'],
 1299: ['217.74.19.0/24', '216.238.144.0/20'],
 8342: ['217.107.200.0/21', '217.107.0.0/18'],
 8359: ['217.74.248.0/21', '217.74.240.0/22'],
 8660: ['213.209.44.0/24', '213.209.40.0/24'],
 13030: ['213.144.144.0/20', '212.51.128.0/19'],
 13414: ['104.244.42.0/24'],
 21232: ['213.160.32.0/19', '195.216.64.0/19'],
 22652: ['216.113.112.0/24', '216.113.104.0/22'],
 59414: ['185.98.120.0/24', '185.98.121.0/24']}

```

Thanks to this information, it will be easier to use the laboratory and navigate it.

Chapter 6

Experimental Laboratories

In this chapter, I will explain the KathBuilder software's output, discussing into depth into the subdirectories and configuration files. I've decided to focus on developing three experimental laboratories based on the Kathará framework, to show the three main phases of a BGP attack. Each laboratory provides different network scenarios and configurations in order to evaluate the importance of security countermeasures.

The following is a brief description of each laboratory:

1. *Initial State* - reproduces the state of the network before the attack, connections between ASs and BGP session configurations are emulated in a manner close to reality, data flow is routed regularly and the network operates according to individual AS policies
2. *Attack State* - reproduces the network conditions at the time the attack occurred, including the misconfigurations required to initiate it, showing the redirected network flow to the malicious AS
3. *Countermeasures State* - reproduces the state of the network under certain countermeasures, demonstrating that the use of security mechanisms could have nullified the effect of the emulated attack

Each laboratory has the following structure:

- *lab.conf file* - responsible for domain connections of all networks in the system
- *ASs Router startup files* - responsible for routers configuration of the corresponding AS
- *ASs Host startup files* - responsible for network configuration of the hosts belonging to their respective AS
- *ASs directories* - responsible for routers BGP configuration of the corresponding AS

Each AS thus has three devices: a router, which is needed to connect to the other ASs in the network and to implement routing policies, and two end-point hosts belonging to the prefixes announced by the respective ASs.

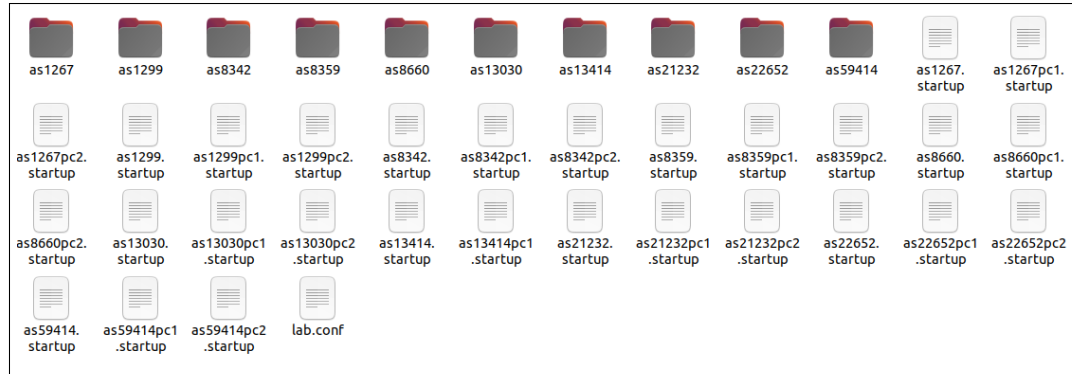
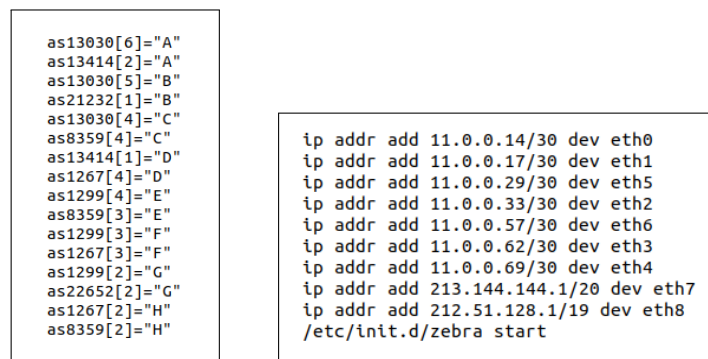


Figure 6.1. Kathará laboratories folder structure.

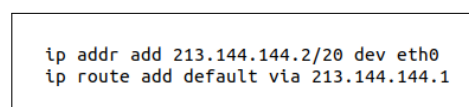
The following three figures show examples of configuration files in the laboratory:

- (a) all couplings between interfaces have been generated within this file so that the networks may communicate with each other
- (b) IP addresses are assigned to all interfaces, and the *zebra* software, which manages TCP/IP routing services, is started
- (c) an IP address is assigned to its only interface that communicates with the router, and it is defined as the default for network access



(a) lab.conf

(b) Router startup

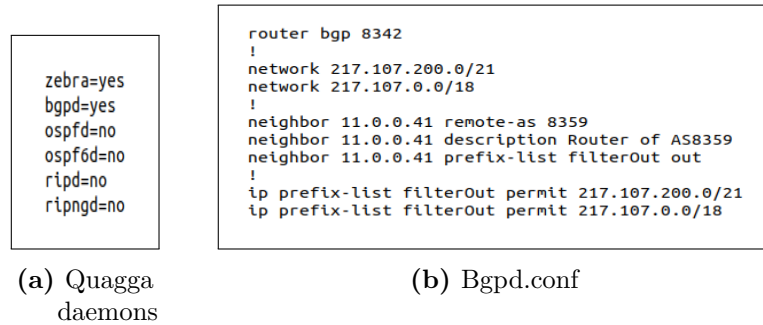


(a) Host startup

Figure 6.3. Basic configuration file examples.

Each AS has a directory dedicated to routing configurations, in our case we are going to configure the Quagga software and define which daemons we want to activate. In the following figures, (a) shows the structure of the Quagga file for the daemons activation, in particular it can be seen that only the *bgpd daemon* was used for these experiments, as it was the only one needed for the purpose.

Figure (b) illustrates an example of BGP configuration in which prefixes are announced, BGP sessions are established with AS neighbors, and inbound and outbound filters are configured.



All of the files mentioned above are part of each laboratory and are automatically created by the KathBuilder software.

In the following paragraphs, we will explore the differences and meaning of the three labs, as well as the tests that were performed on them.

6.1 Initial State

The initial state laboratory, as previously mentioned, emulates the regular behaviour of the network before the incident occurred. The figure below shows the correct operation of the network. Each AS consists of a router and two end-point hosts.

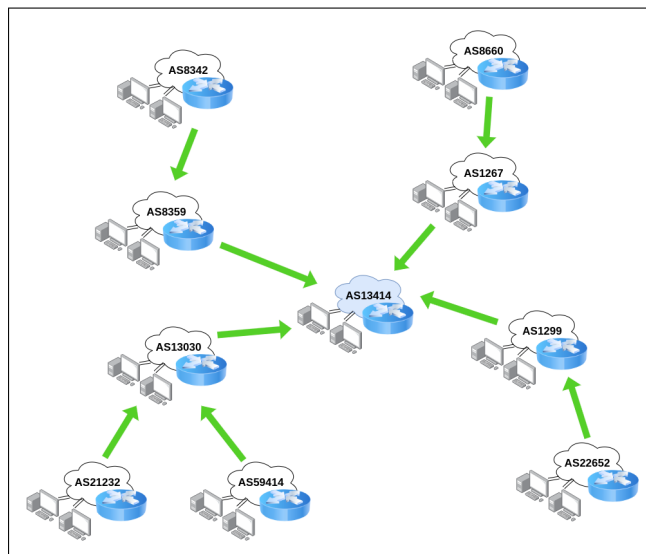


Figure 6.5. Initial State diagram.

Each end-point can communicate with those of the other AS and routers. In the initial state, when an AS wants to communicate with Twitter, it will follow the path shown in the figure. It can be clearly observed how each leaf node relies on its upstream to communicate with the rest of the network.

This laboratory therefore represents the situation in which the Twitter prefix 104.244.42.0/24 is rightfully announced only by AS13414. It is possible to explore the status of connections, best-paths, IP address information and many other details by consulting routers. In particular, the following figure shows the result of consulting the BGP table of AS21232.

```

bgpd> show ip bgp
BGP table version is 0, local router ID is 213.160.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 104.244.42.0/24	11.0.0.53			0	13030 13414 i
* 185.98.120.0/24	11.0.0.53			0	13030 59414 i
*> 185.98.121.0/24	11.0.0.53			0	59414 i
* 185.98.121.0/24	11.0.0.53			0	13030 59414 i
*> 195.216.64.0/19	11.0.0.30			0	59414 i
*> 212.51.128.0/19	0.0.0.0			32768	i
*> 212.141.0.0/16	11.0.0.53			0	13030 i
*> 212.245.0.0/16	11.0.0.53			0	13030 1267 i
*> 213.144.144.0/20	11.0.0.53			0	13030 1267 i
*> 213.160.32.0/19	0.0.0.0			32768	i
*> 213.209.40.0	11.0.0.53			0	13030 1267 8660 i
*> 213.209.44.0	11.0.0.53			0	13030 1267 8660 i
*> 216.113.104.0/22	11.0.0.53			0	13030 22652 i
*> 216.113.112.0	11.0.0.53			0	13030 22652 i
*> 216.238.144.0/20	11.0.0.53			0	13030 1299 i
*> 217.74.19.0	11.0.0.53			0	13030 1299 i
*> 217.74.240.0/22	11.0.0.53			0	13030 8359 i
*> 217.74.248.0/21	11.0.0.53			0	13030 8359 i
*> 217.107.0.0/18	11.0.0.53			0	13030 8359 8342 i
*> 217.107.200.0/21	11.0.0.53			0	13030 8359 8342 i

Displayed 19 out of 21 total prefixes

Figure 6.6. BGP table of AS21232 in the Initial State.

Each line corresponds to a prefix announcement, it is highlighted which one was selected as the best-path, which is the next-hop to cross and, on the right, there's the path in terms of AS.

In this case, we can see that all announcements have the same weight and that there has been no manipulation of special attributes, so the selection of the best path will take place according to the rule of the shortest path.

In the figure below, it is shown the possibility of exploring the log files produced by BGP connections. In this specific example below, AS13414 announces on all its interfaces the BGP update message containing its prefix.

```

root@as13414:/# cat /var/log/zebra/bgpd.log | grep "104.244.42.0/24"
2022/09/22 18:32:13 BGP: 11.0.0.57 send UPDATE 104.244.42.0/24
2022/09/22 18:32:17 BGP: 11.0.0.25 send UPDATE 104.244.42.0/24
2022/09/22 18:32:17 BGP: 11.0.0.9 send UPDATE 104.244.42.0/24
2022/09/22 18:32:17 BGP: 11.0.0.14 send UPDATE 104.244.42.0/24

```

Figure 6.7. BGP update messages of AS13414.

The best paths contained in the BGP routing table are also acquired from the IP routing table, as shown in the figure below.

```

root@as13414:~# ip route
11.0.0.8/30 dev eth3 proto kernel scope link src 11.0.0.10
11.0.0.12/30 dev eth1 proto kernel scope link src 11.0.0.13
11.0.0.24/30 dev eth0 proto kernel scope link src 11.0.0.26
11.0.0.56/30 dev eth2 proto kernel scope link src 11.0.0.58
104.244.42.0/24 dev eth4 proto kernel scope link src 104.244.42.1
185.98.120.0/24 via 11.0.0.25 dev eth0 proto zebra metric 20
185.98.121.0/24 via 11.0.0.25 dev eth0 proto zebra metric 20
195.216.64.0/19 via 11.0.0.25 dev eth0 proto zebra metric 20
212.51.128.0/19 via 11.0.0.25 dev eth0 proto zebra metric 20
212.141.0.0/16 via 11.0.0.14 dev eth1 proto zebra metric 20
212.245.0.0/16 via 11.0.0.14 dev eth1 proto zebra metric 20

```

Figure 6.8. Portion of IP routing table of AS13414.

In the picture we can see that in the IP routing table there are rows acquired by zebra, the software that manages the BGP service. In this laboratory, the relationships between Autonomous Systems were reproduced as closely as possible to reality, implementing the relationships through prefix filtering techniques. Each AS has a list of permitted outbound and inbound prefixes, so that the data flow can be correctly modelled. A vulnerability was left open in the relationship between AS8342 and AS8359, so that the next laboratory could exploit it as it did during the real attack.

6.2 Attack State

The attack phase represents the state of the laboratory in which the vulnerabilities have been exploited and the attack is in place. The incident consists of the announcement of the prefix 104.244.42.0/24 belonging to Twitter, by AS8342, which clearly did not have the authorisation to do that. The announcement originates from AS8342, which uses its upstream AS8359 to spread the malicious message. This action propagated within the network and, due to the lack of proper countermeasures, affected the correct functioning of a group of Autonomous Systems. The diagram of the connections between ASs is shown in the figure below, from which it is possible to observe significant differences from the previous phase.

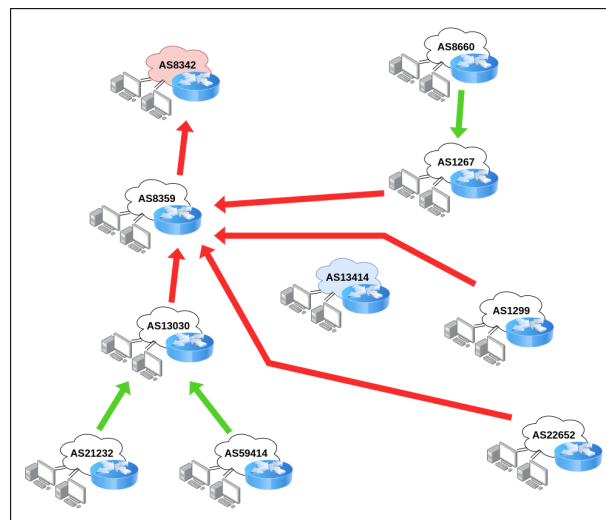


Figure 6.9. Attack State diagram.

Compared to the initial state, most of the routes have been changed. The leaf nodes, the so-called small customers, still refer to their providers to reach the network, while the providers have been diverted to the incriminated Autonomous Systems to reach the Twitter prefix. In this state of the network, anyone attempting to reach Twitter is forced to take the wrong route and reach a false destination. Therefore, in the figure, the Twitter AS is disconnected from the network, as it can no longer be reached by any of its neighbours. This is a serious damage to Twitter's network, which will see its traffic stolen by someone who could potentially inspect confidential information. In a mirror-image fashion, the AS to which the traffic is mistakenly directed could also detect damages, as the heavy traffic could cause service interruptions. In fact, this type of incident is not always caused by hacker attacks, but often by misconfigurations put in place by network operators. The image below depicts the malicious selection of the best path through AS8359 and AS8342, instead of reaching AS13414 as shown in figure 6.6.

BGP table version is 0, local router ID is 213.160.32.1					
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,					
i internal, r RIB-failure, S Stale, R Removed					
Origin codes: i - IGP, e - EGP, ? - incomplete					
Network	Next Hop	Metric	LocPrf	Weight	Path
*> 104.244.42.0/24	11.0.0.29			0	13030 8359 8342 i
* 185.98.120.0/24	11.0.0.29			0	13030 59414 i
*> 11.0.0.66	11.0.0.66	0		0	59414 i
* 185.98.121.0/24	11.0.0.29			0	13030 59414 i
*> 11.0.0.66	11.0.0.66	0		0	59414 i

Figure 6.10. BGP table of AS21232 in the Attack State.

During this experiment, since there were no suitable security mechanisms, announcements propagated in the network and were not filtered properly. This vulnerability caused a chain hijacking of the data flow.

6.3 Countermeasure State

The countermeasures state concerns the situation in which security measures are put into practice and the demonstration of the mitigation of the attack. In this phase, the RPKI framework is active in the laboratory on FRR routers, which use the validator to filter the authorized announcements.

Docker images containing compatible software were used to implement the RPKI framework on the laboratory. Each router was equipped with the quagga service with RPKI support based on RTR communication and Routinator, which runs as a service which periodically downloads and verifies RPKI data. Routers can connect to Routinator to fetch verified data via the RPKI-to-Router (RTR) protocol.

The *Routinator* service was started on routers, where ROAs were downloaded from the Internet allowing it to determine whether a given prefix is valid, invalid or not found. This mechanism works thanks to an association between ASNs and prefixes, which is then cryptographically signed.

If an Autonomous System announces a prefix in the network that it does not own, the router must drop it and not re-announce it on the network.

As an alternative to Routinator, routers can set up a TCP connection to the NameX validator server at address *193.201.40.24* and port *3323*, to obtain the same service as described above. This setting can easily be changed in the RPKI configuration of the router, under the label *cache-server*, where it is also possible to test its connection and read the *prefix-table* containing all downloaded ROAs.

To test the proper functioning of the RPKI filter mechanism, the following test was performed. The Twitter prefix is announced from the AS8342 to its upstream AS8359, which has RPKI validated routers turned off. The result is similar to the previous laboratory, where the advertisement is accepted and propagated in the network, being selected as the best path in the BGP table.

In the figure below, you can see the two announcements received by AS8359, the correct one and the malicious one. Since RPKI is not running, the router does not filter and elects it as the best path.

```
as8359# show ip bgp
BGP table version is 57, local router ID is 217.74.248.1, vrf id 0
Default local pref 100, local AS 8359
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Next hop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
* 104.244.42.0/24  11.0.0.10          0             0 13414 i
*>                11.0.0.34          0             0 8342 i
*> 216.113.104.0/22 11.0.0.17          0             0 22652 i
*> 216.113.112.0/24 11.0.0.17          0             0 22652 i
*> 217.74.240.0/22  0.0.0.0           0            32768 i
*> 217.74.248.0/21  0.0.0.0           0            32768 i
*> 217.107.0.0/18   11.0.0.34          0             0 8342 i
*> 217.107.200.0/21 11.0.0.34          0             0 8342 i
```

Figure 6.11. BGP table of AS8359 with RPKI off.

As we can see in the figure below thanks to the *advertised-routes* option, the Twitter announcement from AS8342 is also announced to the neighbors of AS8359, endangering the proper functioning of the network.

```
as8359# show ip bgp neighbors 11.0.0.73 advertised-routes
BGP table version is 67, local router ID is 217.74.248.1, vrf id 0
Default local pref 100, local AS 8359
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Next hop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*> 104.244.42.0/24  0.0.0.0           0             0 8342 i
*> 216.113.104.0/22 0.0.0.0           0             0 22652 i
*> 216.113.112.0/24 0.0.0.0           0             0 22652 i
*> 217.74.240.0/22  0.0.0.0           0            32768 i
*> 217.74.248.0/21  0.0.0.0           0            32768 i
*> 217.107.0.0/18   0.0.0.0           0             0 8342 i
*> 217.107.200.0/21 0.0.0.0           0             0 8342 i
```

Figure 6.12. Advertised routes of AS8359 with RPKI off.

To mitigate this type of attack, RPKI is a valuable ally. The following figures show how, after turning on the RPKI service, the malicious announcement is discarded upstream and is not re-announced to the other networks.

```
as8359# show ip bgp
BGP table version is 47, local router ID is 217.74.248.1, vrf id 0
Default local pref 100, local AS 8359
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*> 104.244.42.0/24 11.0.0.10         0             0 13414 i
*> 216.113.104.0/22 11.0.0.17         0             0 22652 i
*> 216.113.112.0/24 11.0.0.17         0             0 22652 i
*> 217.74.240.0/22 0.0.0.0           0             0 32768 i
*> 217.74.248.0/21 0.0.0.0           0             0 32768 i
*> 217.107.0.0/18  11.0.0.34         0            10 0 8342 i
*> 217.107.200.0/21 11.0.0.34         0            10 0 8342 i
```

Figure 6.13. BGP table of AS8359 with RPKI on.

```
as8359# show ip bgp neighbors 11.0.0.73 advertised-routes
BGP table version is 77, local router ID is 217.74.248.1, vrf id 0
Default local pref 100, local AS 8359
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*> 104.244.42.0/24 0.0.0.0           0             0 13414 i
*> 216.113.104.0/22 0.0.0.0           0             0 22652 i
*> 216.113.112.0/24 0.0.0.0           0             0 22652 i
*> 217.74.240.0/22 0.0.0.0           0             0 32768 i
*> 217.74.248.0/21 0.0.0.0           0             0 32768 i
*> 217.107.0.0/18  0.0.0.0           0            10 0 8342 i
*> 217.107.200.0/21 0.0.0.0           0            10 0 8342 i
```

Figure 6.14. Advertised routes of AS8359 with RPKI on.

With these experiments, it was confirmed that such attacks would be stopped upstream if the RPKI countermeasure were used by most Autonomous Systems. In this laboratory, for example, it is enough to apply the countermeasure to a single router to be able to completely mitigate the attack. It is very important for the quality of the internet flow that transit networks and networks with many peers implement this security system.

It is important to reiterate that RPKI only mitigates certain types of attacks, such as prefix hijack, and that for complete coverage different security mechanisms should be applied.

Conclusion

The study of the BGP protocol, its characteristics and vulnerabilities, led me to carry out this thesis work which was for me a source of interest, personal and educational growth of great value. I was able to learn and investigate new topics and to approach the world of industry. The initial phase of approaching the protocol was of particular importance, which laid a solid foundation and prepared me adequately for the work to be faced. During this study I have deepened the functioning of the BGP protocol, its basic mechanisms, the characteristics of its messages and the ways to manipulate them, the history of BGP with its vulnerabilities at the base. Being a Cybersecurity student, I paid particular attention to the security aspect, studying the types of attacks and the contexts in which they occur, then focusing on some of the related countermeasures. It was of great interest to study some of the attacks on the BGP protocol, considering one past and one modern. The attacks on YouTube and Twitter are separated by 15 years of difference, but the mechanisms used are always the same. If it was previously impossible to fully protect systems from these attacks, nowadays there are reliable tools able to mitigate them with great success. It is therefore the responsibility of the individual deciding to adopt them, considering that this responsibility affects the quality of the flow of the entire Internet network. We are used to seeing the Internet as one big element, forgetting that it is microscopically composed by each of us. Our virtual behavior affects the entire network, in a positive or negative way. Deciding to adopt security systems to protect your network should be a must, yet most Autonomous Systems lack in this point of view. During this thesis I was able to understand the reasons why many network operators have not yet adopted certain security measures, such as RPKI or BGPsec, examining them with critical but at the same time understanding eyes. Some security mechanisms bring with them very complex implementation structures, expensive to implement, which may require specialized technical experts and dedicated hardware. In the past there has been little awareness on the issue of security, which is why many operators do not believe it is strictly necessary or are not very sensitive to the issue. A great deal of work in this direction is being carried out by the MANRS (Mutually Agreed Norms for Routing Security) initiative, which aims to fill this dangerous gap. It is always important to keep in mind that, for the most complete protection possible, a combination of different countermeasures is required to protect against various types of attacks. The use of KathBuilder software was very useful, as it allows learning and experimenting with BGP networks at an educational level. Having the ability to automatically generate networks and configuration files without having to go into technical implementation details, provides the user with full focus on the study of the network. I like to think that I have created a bridge between the BGPlay widget and

the Kathará framework, allowing user to fully immerse in the network incident that want to analyze. Having the opportunity to reproduce an old accident in a sandbox environment can leave room for experimentation with new solutions and creativity. Finally, I want to look to the future and the next steps to be taken to carry out this project. It would be important to equip it with a Graphical User Interface for a more user-friendly use, and to implement new security mechanisms available within Kathará. It would be very interesting to create a tool for statistical analysis based on laboratory experiments, just as it would be interesting to explore the thorny issue of the definition of the relationship between the Autonomous System. The technological world is constantly changing, and it is important that its users can always make the most of the great opportunity it has offered to humanity.

Appendix A

Twitter's Prefix Hijack Autonomous Systems Details

Below are provided the details obtained through whois queries to the related Autonomous Systems and the use of bgp.he.net.

The information supplied here should not be considered exhaustive or permanent, as they are subject to change, and there is no official dictionary holding certified data.

<u>ASN</u>	6720
<u>Owner</u>	Magwien
<u>Details</u>	<p>Its upstreams include ACONET, A1 Telekom Austria AG.</p> <p>Peerings a Vienna Internet eXchange (VIX) with many Autonomous System including Netflix, GTS Telecom, Hurricane Electric, Cloudflare, Azista, Ripe Ncc, Core Backbone and others. It has around 150 peers.</p>

<u>ASN</u>	8447
<u>Owner</u>	A1 Telekom Austria AG
<u>Details</u>	<p>Peering with over 1000 AS, including Telecom Italia,</p> <p>TATA COMMUNICATIONS (AMERICA) INC, Zayo Bandwidth, MTS, WIND and Twitter. It is certified as Tier 4.</p>

<u>ASN</u>	8607
<u>Owner</u>	Digital Space Group Limited
<u>Details</u>	Peering with around 150 Autonomous Systems, including Level 3, NTT America, MTS, Wind, Sky UK, Facebook, Netflix, Apple and Cloudflare.



<u>ASN</u>	2914
<u>Owner</u>	NTT America, Inc.
<u>Details</u>	Tier 1 peering with more than 1500 Autonomous System.



<u>ASN</u>	9304
<u>Owner</u>	HGC Global Communications Limited
<u>Details</u>	Peering with more than 1000 Autonomous Systems, including Hurricane Electric, GTT Communications, Core-Backbone GmbH, MTS, Cloudflare, TELECOM ITALIA SPARKLE, NTT America and Orange. It is an important Tier 2.



<u>ASN</u>	15435
<u>Owner</u>	DELTA Fiber Nederland
<u>Details</u>	It offers transit to Cogent and GTT Communications. Peering with almost 600 Autonomous System. It's transit downstream are DELTA Zakelijk and Gemeente Vlissingen. It has private peer re- lationship with Google, Microsoft, Amazon, Facebook, Netflix, Twitch and many other.

<u>ASN</u>	19151
<u>Owner</u>	BroadbandONE, LLC
<u>Details</u>	Peering with 400 Autonomous Systems, including NTT America, Hurricane Electric, Zayo, MTS, Level 3 and HGC Global Communications.



<u>ASN</u>	21232
<u>Owner</u>	Genossenschaft GGA Maur
<u>Details</u>	Peering with more than 200 Autonomous Systems. Its upstream providers are Init7 and NTS Workspace. Its customer are Stadtantennen AG and Regio Energie Amriswil.



<u>ASN</u>	13030
<u>Owner</u>	Init7
<u>Details</u>	Transit Autonomous System with almost 1000 peers, including NTT America, Hurricane Electric, Zayo, MTS, Telia Company and Twitter.



<u>ASN</u>	22652
<u>Owner</u>	Fibrenoire Inc.
<u>Details</u>	Peering with almost 500 Autonomous Systems. It is a Tier 2.

ASN 1299
Owner Arelion (Telia Company)
Details Peering with almost 3000 Autonomous Systems.
It is an important Tier 1.



ASN 8660
Owner Italiaonline S.P.A.
Details Peering with almost 300 Autonomous Systems,
mostly italians.



ASN 1267
Owner Wind TRE S.P.A.
Details Peering with more than 1500 Autonomous Systems, including
Telia Company, Level 3, Zayo, Hurricane Electric, MTS, Twitter,
Google, UNIDATA, Facebook, Vodafone and Apple.



<u>ASN</u>	25091
<u>Owner</u>	IP-Max SA
<u>Details</u>	<p>Peering with more than 2300 Autonomous Systems.</p> <p>Its upstream providers are Level 3, Telia Company, GTT Communications, TATA Communications, Orange, Telecom Italia Sparkle. Its customers are Saitis Network, ICANN, Thomas Kernen, Penta, Job-Cloud, France IX Services SASU, Fort IT Services and many others. It is a Tier 3.</p>



<u>ASN</u>	28917
<u>Owner</u>	Fiord Networks
<u>Details</u>	<p>Peering with more than 500 Autonomous Systems.</p> <p>Its upstream provider are GTT Communications. Its customers are Institute for Information Transmission Problems of the Russian Academy of Sciences of A.A. Kharkevic, Scientific-Production Enterprise Business Sviaz Holding LLC, Digit One LLC, KB Rubin Ltd and many others.</p>



<u>ASN</u>	41095
<u>Owner</u>	IPTP LTD
<u>Details</u>	Peering with almost 850 Autonomous Systems.

<u>ASN</u>	3356
<u>Owner</u>	Level 3 Parent
<u>Details</u>	Peering with more than 6400 Autonomous Systems. Its is an important Tier 1.



<u>ASN</u>	51185
<u>Owner</u>	Onecome Global Communications LTD
<u>Details</u>	Peering with more than 2850 Autonomous Systems, including Level 3, Cogent, Hurricane Electric, Telecom Italia Sparkle, MTS, Init7, Fastweb and A1 Telekom Austria AG.



<u>ASN</u>	56665
<u>Owner</u>	Proximus Luxembourg S.A.
<u>Details</u>	Peering with more than 2200 Autonomous Systems, including Hurricane Electric, Zayo, MTS, Fiord Networks, Cloudflare, A1 Telekom Austria AG, Fastweb and Init7. Its upstream providers are Belgacom International Carrier Services, Telia Company, Level 3, NTT America, TATA Communications, Cogent and Voxility.

<u>ASN</u>	57695
<u>Owner</u>	Misaka Network
<u>Details</u>	Peering with more than 2700 Autonomous Systems. It is a transit provider for Datacamp Limited, Global Secure Layer, Voxility, NetActuate, Hurricane Electric, Choopa and Moack.



<u>ASN</u>	60068
<u>Owner</u>	Datacamp Limited
<u>Details</u>	Peering with more than 500 Autonomous Systems, including Telecom Italia Network Sparkle, Core-Backbone, Cogent, Telia Company, GTT Communication, Vodafone Group and MTS.



<u>ASN</u>	33891
<u>Owner</u>	Core-Backbone GmbH
<u>Details</u>	Peering with more than 1700 Autonomous Systems. Its upstream providers are Level 3 and Telia Company. It is an important Tier 2.



<u>ASN</u>	59414
<u>Owner</u>	cloudscale.ch
<u>Details</u>	Peering with more than 200 Autonomous Systems. It is a transit provider for Swisscom, LIG-UPC and Init7. Its customers are Mailxpert, Well-served, Serverbase and Youngsolutions.

<u>ASN</u>	59605
<u>Owner</u>	Mena Levant
<u>Details</u>	Peering with 1250 Autonomous Systems, including Level 3, Telia Company, TATA Communications, Zayo, GTT Communications, MTS, Voxility and Cloudflare.



<u>ASN</u>	137409
<u>Owner</u>	GSL Networks Pty LTD
<u>Details</u>	Peering with almost 3000 Autonomous Systems, including Zayo, Cogent, Hurricane Electric, MTS, Core-Backbone and Fiord Networks.



<u>ASN</u>	199524
<u>Owner</u>	G-Core Labs
<u>Details</u>	Peering with almost 5000 Autonomous Systems. Its upstream providers are Cogent, GTT, TATA, Telia, NTT, MTS, Beeline, Level 3, Telstra, Seacom, China Telecom, Webwerks, TTK and many others.

Appendix B

Results of CAIDA Script Execution

Below are reported the results of the python script 3.1 that provided me with information regarding the relationships between Autonomous Systems based on CAIDA datasets.



AS1267

<u>Providers</u>	1299, 3356
<u>Customer</u>	8660
<u>Peers</u>	25091, 41095, 8607, 60068, 8359, 137409, 15435, 59605, 9304, 56665, 57695, 33891, 199524, 12389, 13030, 13414, 51185, 28917, 8447



AS1299

<u>Providers</u>	-
<u>Customer</u>	25091, 33891, 12389, 13030, 8359, 13414, 60068, 199524, 1267, 59605, 56665, 22652, 8447
<u>Peers</u>	2914, 3356

AS6720

<u>Providers</u>	8447
<u>Customer</u>	-
<u>Peers</u>	33891, 13030

**AS8342**

<u>Providers</u>	12389, 8359
<u>Customer</u>	-
<u>Peers</u>	199524, 51185, 25091, 137409

**AS8359**

<u>Providers</u>	1299, 3356, 12389
<u>Customer</u>	199524, 8342
<u>Peers</u>	137409, 8447, 25091, 33891, 13030, 13414, 41095, 15435, 19151, 51185, 1267, 28917, 59605, 9304, 56665, 22652, 8607

**AS8447**

<u>Providers</u>	1299, 3356
<u>Customer</u>	6720
<u>Peers</u>	137409, 25091, 33891, 12389, 13030, 8359, 13414, 41095, 60068, 15435, 199524, 51185, 1267, 28917, 59605, 9304, 56665, 57695

AS8607

<u>Providers</u>	2914, 3356
<u>Customer</u>	-
<u>Peers</u>	41095, 1267, 13030, 8359

**AS12389**

<u>Providers</u>	1299, 3356
<u>Customer</u>	199524, 60068, 8342, 8359
<u>Peers</u>	2914, 33891, 13414, 13030, 41095, 57695, 1267, 59605, 56665, 8447

**AS13030**

<u>Providers</u>	1299
<u>Customer</u>	21232, 59414
<u>Peers</u>	25091, 41095, 8607, 8359, 6720, 137409, 15435, 19151, 59605, 9304, 56665, 57695, 2914, 33891, 199524, 12389, 13414, 51185, 1267, 28917, 22652, 8447

**AS13414**

<u>Providers</u>	2914, 1299, 3356
<u>Customer</u>	-
<u>Peers</u>	137409, 25091, 33891, 12389, 13030, 8359, 57695, 199524, 15435, 19151, 51185, 1267, 28917, 59605, 9304, 56665, 8447

AS15435

<u>Providers</u>	-
<u>Customer</u>	-
<u>Peers</u>	137409, 25091, 33891, 199524, 13414, 13030, 8359, 51185, 1267, 28917, 56665, 8447

**AS19151**

<u>Providers</u>	2914, 3356
<u>Customer</u>	-
<u>Peers</u>	137409, 33891, 199524, 13414, 13030, 8359, 41095, 51185, 9304, 56665

**AS21232**

<u>Providers</u>	13030
<u>Customer</u>	-
<u>Peers</u>	33891, 25091, 59414

**AS22652**

<u>Providers</u>	1299
<u>Customer</u>	-
<u>Peers</u>	137409, 33891, 25091, 199524, 13030, 8359, 9304, 56665

AS25091

<u>Providers</u>	1299, 3356
<u>Customer</u>	-
<u>Peers</u>	8342, 59414, 60068, 8359, 137409, 15435, 59605, 9304, 56665, 57695, 33891, 199524, 13030, 13414, 21232, 51185, 1267, 28917, 22652, 8447



AS28917

<u>Providers</u>	-
<u>Customer</u>	-
<u>Peers</u>	137409, 25091, 33891, 60068, 13030, 8359, 13414, 41095, 57695, 15435, 199524, 51185, 1267, 59605, 9304, 56665, 8447



AS41095

<u>Providers</u>	3356
<u>Customer</u>	-
<u>Peers</u>	137409, 33891, 199524, 12389, 13030, 8359, 8607, 57695, 19151, 51185, 1267, 28917, 59605, 9304, 56665, 8447

AS51185

<u>Providers</u>	3356
<u>Customer</u>	-
<u>Peers</u>	25091, 41095, 8342, 60068, 8359, 137409, 15435, 19151, 8660, 59605, 9304, 56665, 57695, 199524, 13030, 13414, 1267, 28917, 8447

AS57695

<u>Providers</u>	137409, 60068, 3356
<u>Customer</u>	-
<u>Peers</u>	25091, 33891, 12389, 13030, 13414, 41095, 199524, 51185, 1267, 28917, 59605, 9304, 56665, 8447

AS59414

<u>Providers</u>	13030
<u>Customer</u>	-
<u>Peers</u>	21232, 25091

AS59605

<u>Providers</u>	1299, 3356
<u>Customer</u>	-
<u>Peers</u>	137409, 25091, 33891, 12389, 13030, 8359, 13414, 41095, 57695, 60068, 199524, 51185, 1267, 28917, 9304, 56665, 8447

AS60068

<u>Providers</u>	33891, 2914, 1299, 12389
<u>Customer</u>	57695
<u>Peers</u>	137409, 25091, 199524, 8359, 51185, 1267, 28917, 59605, 9304, 56665, 8447

**AS8660**

<u>Providers</u>	1267
<u>Customer</u>	-
<u>Peers</u>	51185, 199524

**AS33891**

<u>Providers</u>	1299, 3356
<u>Customer</u>	60068
<u>Peers</u>	25091, 41095, 8359, 6720, 137409, 15435, 19151, 59605, 9304, 56665, 57695, 199524, 12389, 13030, 13414, 21232, 1267, 28917, 22652, 8447

AS56665

<u>Providers</u>	2914, 1299, 3356
<u>Customer</u>	-
<u>Peers</u>	25091, 41095, 60068, 8359, 137409, 15435, 19151, 59605, 9304, 57695, 33891, 199524, 12389, 13030, 13414, 51185, 1267, 28917, 22652, 8447

**AS137409**

<u>Providers</u>	-
<u>Customer</u>	57695
<u>Peers</u>	25091, 41095, 8342, 60068, 8359, 15435, 19151, 59605, 9304, 56665, 33891, 199524, 13030, 13414, 51185, 1267, 28917, 22652, 8447

**AS2914**

<u>Providers</u>	-
<u>Customer</u>	60068, 199524, 13414, 19151, 9304, 56665, 8607
<u>Peers</u>	1299, 3356, 12389, 13030

AS3356

<u>Providers</u>	-
<u>Customer</u>	25091, 33891, 12389, 13414, 8359, 8607, 41095, 57695, 199524, 19151, 51185, 1267, 59605, 56665, 8447
<u>Peers</u>	2914, 1299

**AS9304**

<u>Providers</u>	2914
<u>Customer</u>	-
<u>Peers</u>	137409, 25091, 33891, 60068, 13414, 13030, 8359, 41095, 57695, 199524, 19151, 51185, 1267, 28917, 59605, 56665, 22652, 8447

**AS199524**

<u>Providers</u>	2914, 12389, 8359, 1299, 3356
<u>Customer</u>	-
<u>Peers</u>	25091, 41095, 8342, 60068, 137409, 15435, 19151, 8660, 59605, 9304, 56665, 57695, 33891, 13030, 13414, 51185, 1267, 28917, 22652, 8447

Appendix C

Results of Upstream Script Execution

Below are reported the results of the python script 3.2 which processes routeviews collector data to extract information about a specific Autonomous System's upstreams. Since the purpose of my research is to investigate the attack on Twitter, the upstreams that are reported by my script are selected within the circle of Autonomous Systems that have been impacted by the prefix hijack. The output specifies the upstream Autonomous Systems and how many times they are counted by the script. The counter was implemented since not all ASs that appear once as upstream have really that role, therefore it serves as an indicator to evaluate the situation. The small mismatches that there are with the CAIDA dataset method are caused by the fact that the analysis was conducted nowadays and not at the time the attack took place.

AS1267 - Upstreams: 1299: 8, 3356: 11, 13030: 1

AS1299 - Upstreams: 2914: 2, 3356: 8, 13030: 1, 22652: 1

AS2914 - Upstreams: 1299: 5, 3356: 5

AS3356 - Upstreams: 1299: 2, 2914: 2

AS6720 - Upstreams: 8447: 19

AS8342 - Upstreams: 12389: 33

AS8359 - Upstreams: 1299: 2, 3356: 14, 13030: 1, 22652: 1

AS8447 - Upstreams: 1299: 7, 3356: 11

AS8607 - Upstreams: 2914: 8, 3356: 18, 41095: 1

AS8660 - Upstreams: 1267: 16

AS9304 - Upstreams: 2914: 15
AS12389 - Upstreams: 1299: 9, 3356: 13
AS13030 - Upstreams: 2914: 1, 22652: 1
AS13414 - Upstreams: 1299: 8, 2914: 6, 3356: 6, 13030: 1
AS15435 - Upstreams: 13030: 1
AS19151 - Upstreams: 2914: 7, 41095: 1
AS21232 - Upstreams: 13030: 17
AS22652 - Upstreams: 1299: 10
AS25091 - Upstreams: 1299: 5, 3356: 7, 22652: 1
AS28917 - Upstreams: 13030: 1
AS33891 - Upstreams: 1299: 2, 3356: 8
AS41095 - Upstreams: 3356: 16, 13030: 1
AS51185 - Upstreams: 3356: 15, 13030: 1
AS56665 - Upstreams: 3356: 13
AS57695 - Upstreams: 13030: 1, 60068: 23
AS59414 - Upstreams: 13030: 8
AS59605 - Upstreams: 1299: 18
AS60068 - Upstreams: 1299: 11, 33891: 6
AS137409 - Upstreams: -
AS199524 - Upstreams: 8359: 6, 12389: 3

Bibliography

- [1] AFRINIC. What is a route object? what is it used for? <https://afrinic.net/support/resource-members/what-is-a-route-object-what-is-it-used-for>.
- [2] Aftab Siddiqui. Bgp, rpki, and manrs: 2020 in review. <https://www.manrs.org/2021/02/bgp-rpki-and-manrs-2020-in-review/>.
- [3] Aftab Siddiqui. Bgp security in 2021. <https://www.manrs.org/2022/02/bgp-security-in-2021/>.
- [4] Aftab Siddiqui. Lesson learned: Twitter shored up its routing security. <https://www.manrs.org/2022/03/lesson-learned-twitter-shored-up-its-routing-security/>.
- [5] Alessandro Improta, Luca Sani. Bgp route leak incident review. <https://www.catchpoint.com/blog/bgp-route-leak>.
- [6] Alexander Azimov. Bgp as_path verification based on resource public key infrastructure (rpki) autonomous system provider authorization (aspa) objects. <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-verification>.
- [7] Alexander Azimov, Eugene Bogomazov, Randy Bush, Keyur Patel, Job Snijders. Verification of as_path using the resource certificate public key infrastructure and autonomous system provider authorization. <https://datatracker.ietf.org/doc/draft-azimov-sidrops-aspa-verification/>.
- [8] Andrei Robachevsky. Rough guide to ietf 102: Internet infrastructure resilience. <https://www.ietfjournal.org/rough-guide-to-ietf-102-internet-infrastructure-resilience/>.
- [9] Andrei Robachevsky, David Christopher. Rfc 7908 defines bgp route leaks. now, how do we prevent them? <https://www.internetsociety.org/blog/2016/06/rfc-7908-defines-bgp-route-leaks-now-how-do-we-prevent-them>.
- [10] ARIN. Route origin authorizations (roas). https://www.arin.net/resources/manage/rpki/roa_request/.
- [11] Barry Greene Ben Maddison Andrei Robachevsky Job Snijders Sander Steffann David Freedman, Brian Foust. *Mutually Agreed Norms for Routing Security (MANRS) Implementation Guide*. 1 edition, 2018.

- [12] Doug Madory. <https://twitter.com/DougMadory/status/1508466367112093709>.
- [13] Hurricane Electric. Hurricane electric internet service. https://bgp.he.net/AS8359#_peers.
- [14] Internet Society. A regional look into bgp incidents in 2020. <https://www.manrs.org/2021/03/a-regional-look-into-bgp-incidents-in-2020/>.
- [15] Johannes Ullrich. Bgp hijacking of twitter prefix by rtcomm.ru. <https://isc.sans.edu/diary/BGP+Hijacking+of+Twitter+Prefix+by+RTComm.ru/28488>.
- [16] K. Sriram, D. Montgomery, D. McPherson, E. Osterweil, B. Dickson. Rf 7908. <https://www.rfc-editor.org/info/rfc7908>.
- [17] Kentik. What is internet peering? <https://www.kentik.com/kentipedia/what-is-internet-peering/>.
- [18] Kevin Meynell. What are routing incidents? (part 4). <https://www.manrs.org/2020/07/what-are-routing-incidents/>.
- [19] Larry Seltzer. Bgp spoofing - why nothing on the internet is actually secure. <https://www.zdnet.com/article/bgp-spoofing-why-nothing-on-the-internet-is-actually-secure/>.
- [20] Lepinski, Turner. An overview of bgpsec. <https://datatracker.ietf.org/doc/html/draft-lepinski-bgpsec-overview>.
- [21] NOCTION. Bgp security: the bgpsec protocol. <https://www.noction.com/blog/bgpsec-protocol>.
- [22] RATOR. https://mobile.twitter.com/Qrator_Radar/status/1448227705581932548.
- [23] RATOR. https://twitter.com/Qrator_Radar/status/1452587538489778180.
- [24] RATOR. https://twitter.com/Qrator_Radar/status/14402752594043412.
- [25] RATOR. https://twitter.com/Qrator_Radar/status/1394736636089483267.
- [26] RATOR. 264462 massive route leak. <https://radar.qrator.net/blog/264462-massive-route-leak>.
- [27] RATOR. As1221 hijacking 266 asns in 51 countries. <https://radar.qrator.net/blog/as1221-hijacking-266asns>.
- [28] RATOR. As42910 leaking hundreds of prefixes, affecting aka-mai and western asia region. <https://radar.qrator.net/blog/as42910-leaking-hundreds-prefixes>.

- [29] RATOR. Born to hijack. <https://radar.qrator.net/blog/born-to-hijack>.
- [30] RATOR. Indian route leak or there and back again. <https://radar.qrator.net/blog/indian-route-leak-or-there-and-back-again>.
- [31] RATOR. Local leak with global effects. <https://radar.qrator.net/blog/local-leak-with-global-effects>.
- [32] RATOR. Massive vodafone india route leak. <https://radar.qrator.net/blog/massive-vodafone-india-route-leak>.
- [33] RATOR. Weekend route leak by as7552. https://radar.qrator.net/blog/weekend-route-leak-as7552_71.
- [34] RIPE NCC. Managing roas. <https://www.ripe.net/manage-ips-and-asns/resource-management/rpki/resource-certification-roa-management>.
- [35] RIPE NCC. What is rpki? <https://www.ripe.net/manage-ips-and-asns/resource-management/rpki/what-is-rpki>.
- [36] RIPE NCC. Youtube hijacking: A ripe ncc ris case study. <https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study>.
- [37] RIPEstat. <https://stat.ripe.net/docs/02.data-api/bgplay.html>.
- [38] Roma Tre. Kathará | lightweight container-based network emulation system. <https://www.kathara.org/>.
- [39] Antonio Prado Tiziano Tofoni, Flavio Luciani. *BGP: dalla teoria alla pratica*. 2 edition, 2022.
- [40] Tiziano Tofoni, Flavio Luciani, Hiba Eltigani. What is bgp prefix hijacking? (part 1). <https://www.manrs.org/2020/09/what-is-bgp-prefix-hijacking-part-1/>.
- [41] Vasco Asturiano. Bgplay integrated in ripestat. <https://labs.ripe.net/author/vastur/bgplay-integrated-in-ripestat/>.