Detecting BGP origin hijacks

Using a filter-based approach

by

C. Veenman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday January 17, 2019 at 10:00 AM.

Student number:	4495705	
Project duration:	October 1, 2017 – January 17, 2019	
Thesis committee:	Dr. C. Doerr,	TU Delft, supervisor
	Prof. dr. ir. R.L. Lagendijk,	TU Delft
	Dr. A.J. Hoogstrate	Netherlands Defence Academy

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Many processes rely on the availability of the Internet. The Border Gateway Protocol (BGP) is widely used for exchanging routing information between routers and is essential for the successful operation of the Internet. Because BGP has not been designed with security in mind, BGP anomalies such as origin hijacks, route leaks, and link failure often occur. This research proposes a detection system for detecting origin hijacks, which is one of the most common anomalies seen on the Internet. Our detection system uses a filter-based approach. Each filter attempts to validate announcements seen by our detection system. Announcements that could not be verified by any filter are seen as origin hijacks. Because of this approach, origin hijacks that would otherwise be missed by other solutions will be detected. We use multiple data sources such as RIR Statistics Exchange Format Listings (RSEF), routing registries, RPKI Route Origin Authorizations (ROAs) and CAIDA's AS relationship dataset in order to validate announcements. Upon running our detection system on 29 days of BGP traffic, we were able to detect 902 origin hijacks. 83% of the detected origin hijacks had a lifespan of fewer than 2.5 hours which strongly suggests that these were undesired announcements.

Preface

Before I started my master thesis I only had a shallow understanding of how the Internet worked at its core. I greatly enjoyed how this project enabled me to dive deeper into the workings of the Internet. When looking back on the process I realize that there are many areas that are still out there waiting to be discovered. Albert Einstein once fittingly phrased it: "The more I learn, the more I realize how much I don't know."

I could not have finished this thesis without the help of my supervisor Christian Doerr. I want to thank him for giving me advice on my work and pointing out areas on which it could be improved. Christian made me strive for quality and guided me in the process of creating an origin hijack detection system. He was always eager to give me new directions I could advance to. Thanks! I also want to thank the other members of my thesis committee, A. Hoogstrate, and R.L. Lagendijk, for taking the time and effort to read through my thesis.

Furthermore, I want to thank Meike van der Kooij and her family for always motivating and supporting me to work hard on my thesis. I am grateful for the enjoyable moments I could study together with you and the relaxing breaks with a good cup of tea. You helped me through tough times and gave me the energy I needed to work hard and enjoy life. I am very grateful for all your love and support over the past years. Without you, I would not have gotten this far.

I am also grateful for the support of Simone, Anand, and Frits. During our weekly meetings, we had great discussions at which constructive feedback was shared. It was always enjoyable to talk about our progression and the challenges that we faced. I also experienced friendship outside our lives as students for which I am grateful. I also want to thank Tim which sacrificed his time more then once by cycling to Bergschenhoek to study together. Thank you for your friendship and all the enjoyable conversations that we had in these years.

Lastly, I want to express my gratitude towards my dad, Peter Veenman, that passed away on 9 April 2016 and my mother. Dad, you taught me that love is the most important thing in life and that I should never give up on loving others. I still find a smile on my face when I think about all the conversations that we had about, what we believed, the ideal world. You shaped me into who I am right now. Mom, I am proud that you have always supported me in the past years even though times were tough. You were always there for me when I needed advice or someone to listen to my story. I am deeply grateful to both of you.

C. Veenman
Delft, 17 January 2019

Contents

1 Introduction

In a world where many processes rely on the availability of the Internet, digital security becomes an increasingly important concern. The *Border Gateway Protocol*, or BGP for short, is the most widely used protocol for exchanging routing information between routers on the Internet. Without this protocol, routers would not know where to send traffic to and communication on the Internet would be impossible. However, BGP version 4 has been in use since 1994 and has not been designed with security as an important concern. Since then, many vulnerabilities have come to light of which some are now commonly exploited on the Internet. This thesis focuses on detecting BGP origin hijacks, which is one of the most commonly seen BGP anomalies on the Internet.

In simple terms, the Internet can be seen as a large network of interconnected routers. Unique IP addresses are used to allow endpoints on the Internet to communicate with each other. Routing protocols are used to exchange routing information between routers. The exchanged information is then used to determine the outgoing link to which traffic for a particular IP address should be forwarded to. A router has *learned* a new route whenever it receives routing information about a new path to a destination. Because storing routing information for every single IP address would require too much memory, IP addresses are grouped into continuous blocks called *IP prefixes*. An example of an IPv4 prefix is 10.0.0.0/24 which describes that the first 24 bits identify the network and the remaining 8 bits define its endpoints. In simple terms the prefix 10.0.0.0/24 describes the IP addresses 10.0.0.0 to 10.0.0.255. Instead of exchanging routing information of individual IP addresses, routing information of IP prefixes is exchanged to reduce storage space and network overhead. In order for the addresses of an IP prefix to become reachable on the Internet, BGP requires that these IP prefixes are *announced* (i.e routing information is sent) to neighboring routers by a router that is connected to this network. Upon receiving routing information for the first time a BGP-enabled router, called a *BGP speaker*, sends this routing information to all neighboring routers as well, thus eventually reaching a state where all routers on the Internet have learned of this IP prefix. IP addresses are managed by the *Internet Assigned Numbers Authority (IANA)* to ensure global uniqueness. IANA assigns IP prefixes to *Regional Internet Registries (RIRs)* which in turn assign them to organizations. An IP prefix is always owned by at most one organization.

The Problem

In BGP, an *origin hijack* can be described as an unauthorized organization announcing an IP prefix that it does not own. It does so by telling neighboring routers using BGP that all traffic for a certain IP prefix should be sent to him instead of the legitimate owner. These routers then propagate this routing information to other routers on the Internet. When a router then determines that this new route is preferred over the old route, it will redirect traffic to the hijacker instead of the legitimate owner. The *BGP decision process*, which is defined as part of the BGP protocol [57], compares the attributes of different routes in order to decide on the most preferred route to an IP prefix. BGP origin hijacks can have a great impact on the stability of the Internet. When prefixes are hijacked, partial or complete connectivity of the victim to the Internet might be lost, as some traffic will not reach the correct destination anymore. A successful origin hijack also allows a hijacker to use the hijacked prefix for any desired purpose, such as sending spam or impersonating as the

legitimate owner as well as performing a *Man in the Middle (MitM)* attack by eavesdropping or manipulating traffic intended for the legitimate owner.

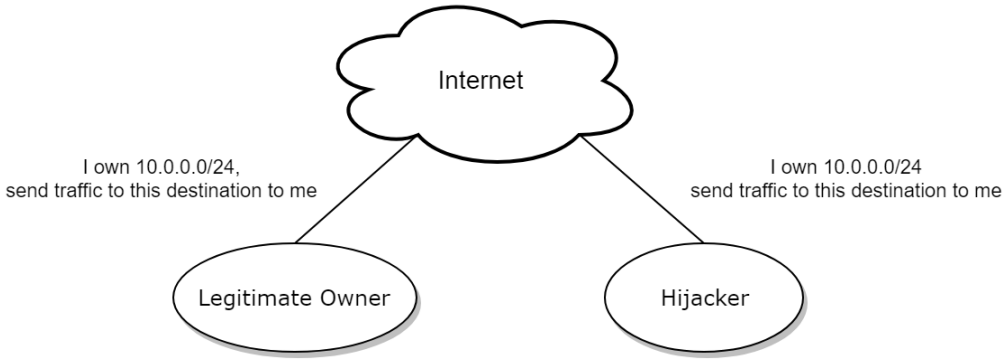


Figure 1.1: A BGP origin hijack

Figure 1.1 shows an example of a origin hijack. Let us suppose an organization is the legitimate owner of prefix 10.0.0.0/24 and that it announces to its neighbors that traffic destined to this destination should be sent to him. These neighbors then forward this route to the rest of the Internet and traffic is being sent to the legitimate owner. When a hijacker tries to perform an origin hijack, it also announces to its neighbors (and indirectly the Internet) that it owns prefix 10.0.0.0/24. Neighbors receiving this announcement now have to choose between sending traffic to the legitimate owner and the hijacker. Based on the heuristics that BGP uses to determine the most preferred path, some routers might now send traffic to the hijacker instead of the legitimate owner. The hijacker can now use these IP addresses for any desired purpose and manipulate the traffic that is now being sent to him instead as he sees fit.

BGP has no knowledge about the ownership of IP prefixes and by default trusts all the announcements that it receives. It does not attempt to validate if the organization that announced an IP prefix, is actually the legitimate owner. Because origin hijacks are easy to perform (one would only need to configure a BGP speaker to announce an erroneous IP prefix) it is one of the most commonly seen BGP anomalies on the Internet. Because BGP itself does not provide any protective measures against origin hijacks, network operators have to resort to external solutions.

Protection against origin hijacks

Origin hijacks have been researched for a long time but no solution to date has been able to successfully stop BGP origins from occurring. The challenge, when faced with the problem of stopping BGP origin hijacks, is mainly due to the decentralized nature of the Internet. Prefixes belonging to an organization can never be fully protected as long as other organizations naively accept announcements from their neighbors. Many solutions that were invented, such as the Internet Routing Registry, BGPSec, and Route Origin Authorizations, all suffer from this problem. As long as organizations refuse to participate in the adoption of protective technologies, they will always be unprotected and might be affected by origin hijacks. As can be seen in Figure 1.2, unprotected organizations that are affected by an origin hijack also affect protected organizations, since traffic destined for a protected organization and originating from an affected organization does not reach the correct destination anymore. Both organizations lose connectivity to each other when either of them has been affected by an origin hijack of the other's prefixes.

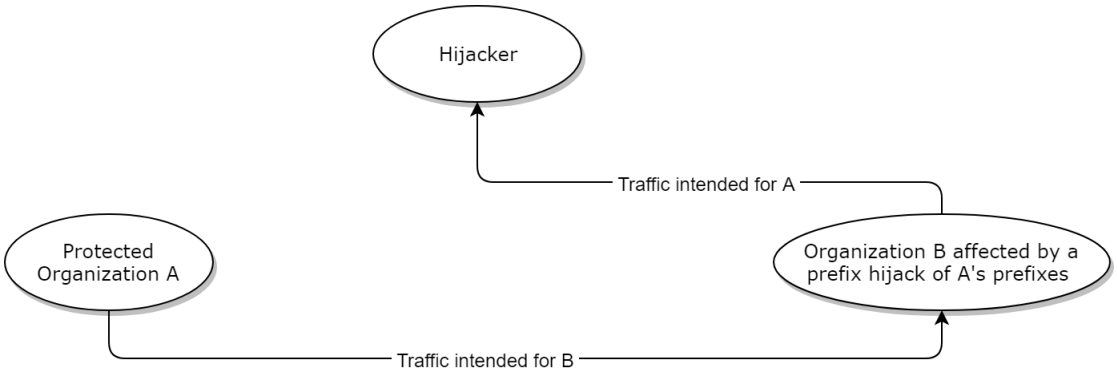


Figure 1.2: Example showing bidirectional impact of unprotected organizations

One of the first attempts to solve this problem was by using *routing registries* that store information about network operators and the configuration of their routers. These routing registries are maintained by network operators themselves and contain data on which organizations are allowed to announce an IP prefix. This information may then be used by BGP speakers to validate incoming announcements. Many providers still require their customers to register their prefixes at a routing registry before they will forward their announcement further on the Internet. However, even though data is stored in a structured language called the *Routing Policy Specification Language (RPSL)* [28], many routing registries do not validate whether the information inserted is correct. For example, many routing registries use different RPSL attributes when describing an object and some routing registries contain duplicate data of which one record is slightly outdated. These quirks make it harder for BGP speakers to validate announcements using a standardized approach. Also, many different routing registries exist and in order to verify an incoming announcement, a BGP speaker would have to query each of them, significantly slowing down the validation process.

One of the latest technologies being used to protect against hijacks are *Route Origin Authorizations (ROAs)* [50]. This technique makes use of cryptographic signatures to sign a statement made by a network operator on which organization is allowed to announce its prefix. Because it is verified and signed by a trusted third party such as a Regional Internet Registry, the likelihood of someone accidentally or intentionally imposing as someone else is very unlikely. As opposed to routing registries, ROAs use a standardized format which makes it easier for BGP speakers to operate on. There are however limitations that currently prevent ROAs from being an effective solution. BGP speakers that use ROAs to validate incoming announcements need more processing power to validate the cryptographic signatures. It also requires the availability of a ROA repository in order to retrieve ROAs for a certain prefix. Another major reason why using ROAs is not yet successful at defending against BGP origin hijacks is related to its adoption. Even though an increasing amount of network operators have created ROAs for their prefixes, they still cover only 11% of the entire IPv4 address space [16]. This would mean that BGP speakers using ROAs to validate prefix announcements are currently only able to protect a small fraction of the entire Internet.

BGPsec, on the other hand, extends BGP with cryptographic security such that the path through which announcements travel cannot be spoofed. This is done by forcing BGP speakers to cryptographically sign the intended receiver of the announcement. Because it can now be proven that the receiver is indeed the intended receiver of the announcement, an attacker would not be able to pretend that it lies on the path from the origin to the destination. If the attacker would be able to fake a shorter path to the destination, this route might be preferred over other routes and traffic might travel to the attacker instead. The fact that BGPsec only focusses on verifying path correctness, makes it incapable of detecting origin hijacks. BGPsec, however, faces some other significant limitations as well. One of its limitations is that once a BGP speaker does not support BGPsec, all BGPsec data will be removed from the announcement beyond this point and BGP speakers that receive an announcement from such a BGP speaker will not benefit from BGPsec anymore. This means that if network operators at key positions in the Internet topology do not support BGPsec, it cannot be fully utilized. Another reason is that using BGPsec increases the size of BGP announcements by a factor of 15 [9]. This significantly increases the amount of memory being used by BGP speakers. Some routers would have to be replaced in order to support BGPsec. The verification of each cryptographic signature in messages using BGPsec also requires a lot of processing power and would significantly increase the strain on BGPsec-enabled routers. All of these reasons significantly decrease the adoption rate of BGPsec and as a result also its ability to protect the BGP ecosystem.

Detection of origin hijacks

Because it is impossible to fully protect against origin hijacks as long as there are organizations that are not adopting protective technologies, many researchers have shifted their focus to the detection of origin hijacks such that, if an origin hijack occurs, a network operator can recover from such an attack as quickly as possible. The detection approaches that have been presented so far can be divided into two categories. Either they make use of historical BGP data or they make use of active probing in order to detect origin hijacks.

One of the latest detection techniques using historical BGP data is ARTEMIS [5]. The system is deployed alongside the BGP speaker(s) that the organization operates. The organization has to manually configure which origin AS is allowed to announce their prefixes. Once a prefix announcement for one of the configured prefixes has been received it is matched against this configuration. Upon detecting a mismatch an alarm is raised. The limitation of this approach is similar to that of techniques that provide protection against origin hijacks, namely that it is only capable of detecting origin hijacks of its own prefixes. This approach, therefore, fails to overcome the shortcomings of protective measures, since other organizations that this organization is communicating with might still be affected. Other approaches using historical BGP data such as [45], [48] and [66] are also not able to detect origin hijacks other than those targeting their own prefixes.

NETREVIEW [41] is the only system to our knowledge that uses historical BGP data and is also capable of detecting origin hijacks that target other organizations. In order for NETREVIEW to work, each network operator has to specify a high-level specification of their routing policies. This specification is then stored in a global database such that other network operators can use this information to verify incoming announcements. Each system using NETREVIEW also has to store logs for at least 1 year. As has been seen with RPKI's adoption, network operators are often reluctant when having to manually create and maintain their routing specifications. The shared database also creates a central point of failure into the system and the requirement of having to store log files for at least one year might be a requirement that cannot be met by the current infrastructure of a network operator. As has BGPSEC and RPKI have already shown, the adoption of new infrastructure changes often is very slow.

Approaches that use reachability checks often make use of active probing [68][43][61][67]. By sending probes, an increased burden is placed on the network, which can decrease the throughput of common traffic. [11] is the only approach we found that passively monitors reachability information in packets instead of sending active probes. It is built on the assumption that a hijacker is at a different location often further away from the legitimate origin. Whenever the Time to Live field inside packets change by a significant amount that is larger than the threshold, an alert will be raised. The problem of this approach is that it is unable to detect all origin hijacks. Cases where the hijacker is equally far away to the origin as the legitimate owner, will not be detected. Approaches such as [11] also face high false positive rates since relocations to different geographical areas will

also trigger alerts. High false positive rates often lead to an unusable system because the maintainers that receive a hijack alert would have to manually inspect each case in order to verify whether it is correct. Other approaches such as [43], [61], and [67] are also confronted with high false positive rates because of link failure.

Many solutions that make use of reachability information often have slow detection times. Often the probe results need to be compared with the results of other vantage points, which often results in detection times in the order of minutes [68][61]. If the victim has to wait till an erroneous announcement reaches its network, the announcement might have already impacted many other networks on the Internet. Detecting an origin hijack as early as possible in time enables the legitimate owner to respond quickly to the incident, and often reducing the impact of the hijack.

Solution

As has been shown, the current state of the art origin hijack detection systems have shortcomings such as slow detection times, the inability to detect origin hijacks other than those targeting our their organization and the large number of false positives that some systems generate. This thesis proposes a system that can detect BGP origin hijacks without these limitations. Our system uses a filter-based approach where each filter attempts to validate a received announcement. We use a variety of data sources to validate announcements, such as the Internet Routing Registry (IRR), RIR Statistics Exchange Format (RSEF) listings, Route Origin Authorizations (ROAs), and CAIDA's AS-relationship dataset. Because only the announcements that are confirmed as valid are excluded, virtually no origin hijacks are missed. One would suspect that many false positives would be generated using our approach but the contrary is true. Our results show that 83% of the detected origin hijacks have a lifespan shorter than 2.5 hours, which strongly suggests that these were unintended announcements. The proposed system is also able to detect origin hijacks in less than a second upon receiving an announcement, unlike [68] and [61] which need minutes to detect origin hijacks. This enables network operators to respond faster and reduce the impact of the prefix hijack. Our system can be deployed at multiple locations in order to detect origin hijacks as early as possible but does not have to compare results with other instances in order to reach a conclusion. Because our algorithm can validate BGP announcements accurately and in real-time, it is a good candidate for blocking the propagation of BGP origin hijacks or placing them in quarantine as has been proposed in [45]. By preventing further propagation of erroneous announcements, the impact of origin hijacks can be partially mitigated. It is also possible to actively recover from classified origin hijacks by announcing a valid route, that was previously announced, back to affected BGP speakers. The earlier this is done, the lower the impact an origin hijack might have.

Outline

The research question that this thesis will aim to answer is formulated as "How to detect origin hijacks accurately in real-time?". In order to answer this question, the thesis is structured as follows: Chapter 2 covers the fundamentals of BGP and BGP related topics such that any reader that does not have a firm knowledge of these topics is able to understand later chapters after reading this. Readers that already have a firm knowledge of BGP and its vulnerabilities can skip this chapter. Chapter 3 will cover the current state of the art methods on detecting origin hijacks and will show their limitations. This chapter will end with an explanation of how our solution improves on the limitations of other approaches. Chapter 4 will explain the data sources that are being used in our detection algorithm. Chapter 5 describes our detection algorithm in detail and explains how it can be used to detect origin hijacks. Chapter 6 explains the results of running our detection algorithm on 29 days of BGP announcements. Our conclusion and discussion based on these results will be described in Chapter 7.

2
Background

In order to understand the concepts used in subsequent chapters, it is necessary to obtain sufficient background knowledge. Readers that already have sufficient knowledge about BGP and its related topics can skip this chapter. Section 2.1 will explain the foundation that is required to understand the Border Gateway Protocol. Concepts such as Autonomous Systems and IP prefixes are covered here, as well as the organizations that manage these Internet resources such as IANA and the five Regional Internet Registries (RIRs). Section 2.2 contains a short introduction of BGP, that covers the essential parts used in this thesis. For more in-depth knowledge on BGP the reader is referred to [57]. Section 2.3 explains the most commonly seen routing policies on the Internet. Section 4.6 covers the Internet Routing Registry (IRR). Section 2.4 concludes this chapter with a discussion on commonly seen BGP anomalies. We assume that the reader is familiar with network fundamentals such as IPv4 and IPv6 addresses and IP subnets.

2.1. Fundamentals

The Internet is the global system of interconnected routers that use the Internet Protocol Suite (also known as TCP/IP). In order for a router to correctly send packets to the right destination (identified by an IP address), routers need to exchange routing information to find a path to the destination. An *Exterior Gateway Protocol (EGP)* is used to exchange routing information between the routers of different organizations. The *Border Gateway Protocol (BGP)* is the de-facto Exterior Gateway Protocol being used on the Internet and therefore serves a critical function in today's Internet infrastructure. BGP can also be used as an *Interior Gateway Protocol (IGP)*, meaning that it is used for exchanging routing information inside an organization's network. When BGP is used as an IGP it is called iBGP whereas it is called eBGP when used as an EGP. However because our focus lies on detecting origin hijacks, we will only cover BGP as an EGP. Whenever BGP is mentioned the reader can assume we mean eBGP unless explicitly stated otherwise. In order to understand how BGP works and where its weaknesses lie, it is necessary to understand the concept of IP prefixes and Autonomous Systems (ASes).

2.1.1. IP Prefixes

Users of the Internet can communicate with each other by using IP addresses. It is however impossible for routers to exchange and store routing information for each IP address because routers would then need a very large amount of memory. In order to solve this problem, IP addresses are grouped together into an *IP prefix*, which is a continuous collection of IP addresses. An example of an IPv4 prefix is 123.43.12.0/24, which tells us that the first 24 bits of this address are part of the prefix and that the remaining 8 bits can be used to form actual addresses that can be used to communicate with other IP addresses on the Internet. 123.43.12.0/24 is used to specify the addresses 123.43.12.0 to 123.43.12.255. In order for a router to know to which interface traffic should be forwarded, routing information about these IP prefixes are exchanged using the BGP protocol. The BGP protocol then uses the obtained routing information to choose a path to the destination IP address. How the preferred path is chosen will be covered in Section 2.2.4.

Currently, two versions of the IP protocol exist. IPv4 which consists of addresses that are 32 bit in size, and IPv6 which consists of addresses that are 128 bits long. IPv6 was invented because the amount of unused

IPv4 addresses was depleted. Even though IPv6 is gaining adoption, IPv4 is still the most dominant version on the Internet. IPv6 prefixes work the same way as IPv4 prefixes but often have a larger prefix size due to the increased amount of bits that may now be used.

2.1.2. Autonomous Systems

BGP exchanges routing information between *Autonomous Systems (ASes)*. An AS is a connected group of one or more IP prefixes run by one or more network operators which have a single and clearly defined routing policy [42]. Typically each network operator has a single AS and is assigned a globally unique *Autonomous System Number (ASN)*. These Autonomous System Numbers (ASN) are used by BGP to determine the preferred path to the destination as well as for avoiding routing loops. Each AS can connect to other ASes to exchange routing information, this is called *peering*. The AS with whom such a relation is formed is called a *peer*. The concept of peering should not be confused with a Peer to Peer relationship which is a specific type of peering relationship. Peer to Peer relationships will be covered in Section 2.3. In order for an organization to be connected to the rest of the Internet, an AS needs to be connected to one or more providers. A *provider* is a network operator that provides customers with access to the rest of the Internet by propagating the customer's traffic further on the Internet and sends incoming traffic, destined for the customer AS, to the customer. Typically a customer has to pay for this service. Which routing information is exchanged over a peering relationship can be controlled by configuring *routing policies* on the participating routers. Routing policies will be covered in Section 2.3.

2.1.3. Autonomous System Number and IP prefix allocation

In order to make sure that IP addresses and ASN are globally unique the *Internet Assigned Numbers Authority (IANA)* was formed. IANA is in charge of managing all Internet resources such AS numbers and IPv4 and IPv6 addresses. It does so by allocating the resources to *Regional Internet Registries (RIRs)* which in turn assign the resources to other organizations. Instead of allocating IP prefixes directly to an organization, a RIR can also choose to delegate a prefix to a *Local Internet Registry (LIR)* or *National Internet Registry (NIR)*. A Local Internet Registry is an organization that is in charge of the delegation of prefixes to their customers. National Internet Registries are organizations that manage the prefixes for organizations in their designated country. An example is JPNIC, which is the National Internet Registry for Japan.

Regional Internet Registries

There are currently five RIRs that each serves their particular area: ARIN for North America, LACNIC for South America, RIPE for Europe, Russia and Greenland, APNIC for Asia and Australia, and AFRINIC for Africa. Each of these RIRs is tasked with allocating Internet resources to their specific area. RIRs also participate in the *Number Resource Organization (NRO)*, which undertakes joint activities to improve the BGP ecosystem. AFRINIC became operational in 2005 and has since then taken over Africa from RIPE. It is interesting to note that ARIN is assigned 45% of all the IPv4 addresses in the world, followed by 24% to APNIC and 22% to RIPE [25]. This is mainly due, at the early rise of the Internet, many /8 blocks were assigned to organizations such as the US Department of Defence, US Postal Service which fell under ARIN's authority. Only later when Internet usage became more prevalent, the exhaustion of IPv4 addresses became a problem.

2.2. The Border Gateway Protocol

In this section, a short introduction to the Border Gateway Protocol will be given. For the complete protocol specification, the reader is referred to [57]. BGP is used to exchange routing information between BGP speakers. A *BGP speaker* is router that uses the BGP protocol. BGP version 4 which is the latest version of the BGP protocol has been standardized since 2006 but in practice, the protocol has already been in use since 1994. The BGP protocol is a relatively simple protocol that only has four type of messages, namely OPEN, KEEPALIVE, UPDATE, NOTIFICATION. All BGP messages have a common header containing the length and type of the message. This header is necessary to distinguish between the different types of BGP messages. Section 2.2.1 will explain how a BGP peering session is created and managed using the OPEN, KEEPALIVE, and NOTIFICATION message types. Section 2.2.2 will explain how routing information is exchanged using UPDATE message type. Section 2.2.4 will explain how BGP decides on the preferred route to a certain destination. Section 2.2.5 will cover extensions to the BGP protocol that are relevant to this thesis.

2.2.1. Establishing and maintaining a peering connection

In order to exchange routing information, two BGP speakers need to set up a peering session. Both BGP speakers open up a TCP connection on port 179. When a TCP connection has been opened both BGP speakers send an OPEN message. The format of an OPEN message can be seen in Table 2.1.

Field	**Size in bytes**	**Description**
Version	1	The version of the protocol, currently 4.
My ASN	2	The ASN of the BGP Speaker sending this OPEN message.
Hold Time	2	The maximum amount of seconds between messages
BGP Identifier	4	The BGP identifier of the BGP speaker.
Optional Parameter Length	1	The size in bytes of the Optional Parameters field.
Optional Parameters	Variable	Extra capabilities can be negotiated here.

Table 2.1: BGP OPEN Message Format

The ASN field is used to verify which ASN the receiving BGP speaker is talking to and can possibly close the connection if it was not configured to peer with this AS. The Hold Time field contains the maximum amount of seconds that may elapse between UPDATE and KEEPALIVE messages. The minimum of both OPEN messages is chosen. If at any time a message has not been received in this time window the connection is terminated and has to be reconstructed again. This is done such that, may link failure occur, the program is able to detect this and may try to re-establish the session. A KEEPALIVE message has no body and is simple a BGP message header with the type field set to KEEPALIVE. A KEEPALIVE message is sent in order to prevent the connection from timing out due to the hold timer. The Optional Parameter field is used to negotiate extra parameters and extensions for the connection such as whether to use Multiprotocol Extensions and 4-byte ASN numbers. Section 2.2.5 will elaborate more on commonly seen BGP extensions.

In case any error occurs during the BGP peering session, a NOTIFICATION message is sent. These NOTIFICATION messages can help diagnose the problem. After a NOTIFICATION message has been sent, the connection should be closed immediately. The data sent with a NOTIFICATION message is specific to the error code of the NOTIFICATION message. It is mainly used for diagnostic purposes and often contains a string explaining the error. For example "Hold Timer has been expired." to indicate that the peer has not received a message within the specified hold time.

2.2.2. Exchanging routing information

Routing information is exchanged using UPDATE messages. The UPDATE message format is described in Table 2.2.

Field	Size in bytes	Description
Withdrawn Routes Length	2	Specifies the length in bytes of the Withdrawn Routes field.
Withdrawn Routes	Variable	Contains prefixes that are to be withdrawn.
Path Attribute Length	2	Specifies the length in bytes of the Path Attribute field.
Path Attributes	Variable	Contains meta data of the prefixes to be updated.
NLRI	Variable	Contains the prefixes that are to be updated.

Table 2.2: BGP UPDATE Message Format

Routing Information Bases

Each BGP speaker stores its routes and associated path attributes in a *Routing Information Base (RIB)*. A RIB consists of a Adj-RIBs-In, Loc-RIB, and the Adj-RIBs-Out. The Adj-RIBs-In contains the routing information learned from incoming UPDATE messages that were received from other BGP speakers. These routes are not yet accepted as the best route but can be used in the decision process as a potential best route. The Loc-RIB contains the routes that are accepted as the best route and will be used by the BGP speaker to forward traffic. The Adj-RIBs-Out contains the routes that are to be advertised to other peers.

Once a new peering session is constructed both BGP speakers exchange the entire contents of their Loc-RIB with each other. Each time a new UPDATE message is received, it is placed in the Adj-RIBs-In and upon selected as the best path, in Loc-RIB and Adj-RIBs-Out respectively. However, BGP speakers can be configured in a variety of ways which might prevent the propagation of UPDATE messages to some peers. By configuring a BGP speaker, routing policies can be constructed. Routing policies will be covered in Section 2.3.

In order to choose the best path, extra information is contained in the *Path Attribute* field. This information applies to all prefixes that are contained in the *NLRI (Network Layer Reachability Information)* field. The Withdrawn Routes field contains the prefixes that should be removed from the Loc-RIB, such that traffic for these prefixes cannot travel via this peer anymore. An UPDATE message containing withdrawn routes that were previously in the Loc-RIB are also to be propagated to peers to inform them that it is no longer a valid path. If a route that was currently the best route has been withdrawn, the decision process will run again such that another route that was previously not the most preferred route, can now be used as the preferred route. If such a route is not available, traffic will be dropped.

SOAS and MOAS

It is possible that a prefix is announced by multiple ASes at the same time. This is called a Multiple Origin AS (MOAS) whereas a prefix being announced by a single origin is called a Single Origin AS (SOAS). A MOAS might occur, for example, when an AS is privately peering with two providers that both announce its prefix, or when anycast is deployed between multiple ASes.

2.2.3. Path Attributes

In this section, the path attributes that are described in the original BGP specification will be explained. Common extensions to the BGP protocol that include extra path attributes will be covered in Section 2.2.5.

AS_PATH

The AS_PATH attribute is a mandatory attribute that should be included in every UPDATE message that contains prefixes in its NLRI section. Each AS_PATH attribute consists of path segments which are either of type AS_SET or AS_SEQUENCE. The AS_PATH attribute represents the path of ASes that traffic should travel through in order to reach the destination. An example AS_PATH belonging to an UPDATE message containing prefix 145.94.0.0/16 might be (AS9304, AS1103, AS1128). This means that AS1128 (owned by TU Delft) is announcing the prefix 145.94.0.0/16 and that in order for traffic to reach AS1128 it has to travel through AS9304 and AS1103 respectively. The AS_PATH attribute is used in the decision process to select the most optimal path and is used to detect routing loops as well. Every time a BGP speaker receives an UPDATE message containing its own AS in the AS_PATH attribute, the UPDATE message is discarded since it has already received

this route previously. Each time a BGP speaker sends a BGP UPDATE message to a peer it should prepend its own AS number to the AS_PATH. The AS_SET segment type is used when BGP speakers merge multiple prefixes with similar attributes together to form one larger prefix. In this case, the AS_PATHs cannot always be merged together, and an AS_SET segment is used to indicate that traffic travels through one or more ASes contained in the set. As will be shown in Chapter 6, AS_SETs are not used often anymore. For more details on the process of merging AS_PATHs, see RFC[57].

NEXT_HOP
The NEXT_HOP attribute should be included in all UPDATE messages unless the UPDATE message contains only withdrawn routes. It specifies the IPv4 address that should be used as a next hop to the prefixes contained in the NLRI field. In other words, when a route is accepted, all traffic is forwarded to the interface that knows how to reach this next hop IPv4 address. It is important that the next hop should be reachable by using the current Loc-RIB otherwise the BGP speaker would not know which interface to use when forwarding traffic. Therefore the IP address contained in the NEXT_HOP attribute is often the IP address of an interface that is directly connected to the interface

ORIGIN
The ORIGIN field is a mandatory field that is generated by the BGP speaker that originated the UPDATE message. It specifies how the route was generated and can either be EGP, IGP or INCOMPLETE. It is set to INCOMPLETE when it cannot be determined anymore because of, for example, aggregation.

LOCAL_PREF
LOCAL_PREF is used to express a preference for a given route over other routes. LOCAL_PREF is only used on iBGP peering sessions. It should not be confused with the administrative weight property which can be set by administrators to express a preference for a given route.

MULTI_EXIT_DISC
Multiple Exit Discriminator is an optional non-transitive BGP attribute that is intended to be used with external BGP peering relationships. When an AS has multiple links to a peering AS the path with a MULTI_EXIT_DISC set to the lowest value will be preferred. This attribute should not be propagated to other peers.

ATOMIC_AGGREGATE
When a BGP speaker decides to aggregate prefixes into one larger prefix and does not include all ASes in the AS_SET property, it should include this attribute. This attribute should be propagated to other peers as well.

AGGREGATOR
An AGGREGATOR attribute may be included when a BGP speaker decides to aggregate prefixes into one larger prefix. It includes its own AS and its IP address. The attribute should be propagated to other peers as well.

2.2.4. Decision Process
In order to select the best path to a certain destination, each prefix in the Adj-RIBs-In goes through a decision process which determines if it can be included in the Loc-RIB and Adj-RIBs-Out. The decision process consists of a series of steps which express the preference in order. The conditions are expressed in Table 2.3.

Condition	**Description**
NEXT_HOP	If the NEXT_HOP is not reachable, discard this route from the decision process.
Administrative	Select the route with the highest (manually set) administrative weight.
LOCAL_PREF	Prefer the route with the highest LOCAL_PREF.
Locally Generated	If a route is locally generated (such as an aggregate) prefer this route.
AS_PATH Length	Prefer shorter AS_PATH length. An AS_SET counts as 1 in length.
ORIGIN	Prefer EGP over IGP and IGP over INCOMPLETE.
Multi Exit Descriminator	Choose the path with the lowest Multi Exit Descriminator value.
EGP over IGP	Prefer routes learned by an EGP connection over those learned from an IGP.
Lowest IGP metric to NEXT_HOP	Choose the path with the lowest IGP value to the NEXT_HOP.
Oldest Route	Select the oldest route, that is, the route that was learned first.
Lowest Router ID	Selects the path of the peer with the lowest router ID.
Lowest Interface IP address	Select the route that was received on the lowest inteface IP address.

Table 2.3: The BGP Decision Process

It is important to note that the decision process works with prefixes that are exactly the same. It often occurs that more specific prefixes are also announced. When relaying traffic, more specific prefixes are always preferred over less specific ones. However less specific prefixes will be installed in the Loc-RIB and Adj-RIBs-Out if they are considered the preferred route according to the decision process. Upon determining a new preferred route, the route is placed in the Adj-RIBs-Out and is propagated to its peers as well.

2.2.5. BGP Extensions

BGP has been designed with extensibility in mind. By exchanging capabilities in the OPEN message the behavior of the BGP protocol can be modified or extended. This section will cover the BGP extensions that are relevant when reading other chapters. For a list of all BGP capabilities, the reader is referred to [10].

Multiprotocol Extensions for BGP-4

Originally BGP was only designed to be used with IPv4 addresses. Since there are also other protocols other then IPv4, such as IPv6 and IPX, BGP had to be extended in order to cope for these other protocols as well. The Multiprotocol Extensions for BGP-4 cover this problem [30]. The Multiprotocol Extensions introduce two new path attributes called Multiprotocol Reachable NLRI (MP_REACH_NLRI) and Multiprotocol Unreachable NLRI (MP_UNREACH_NLRI). The MP_REACH_NLRI can carry NLRI information about a variety of protocols and can also include a NEXT_HOP address of a protocol different than IPv4. MP_UNREACH_NLRI can carry withdrawn routes of a variety of protocols. Both of these path attributes are used to exchange IPv6 routing information between BGP speakers. If a BGP speaker does not understand the Multiprotocol Extensions these attributes are not propagated to other peers. For details about the format of these attributes see [30].

4-byte ASN capability

The original specification of BGP only allowed for a 2-byte ASN. However since the rise of the Internet demanded more numbers than the 2-byte pool could initially provide, this had to be extended to 4 bytes. The Extended 4-byte ASN capability solves this problem [63]. By exchanging this BGP capability in the OPEN message a BGP speaker indicates that all ASN fields should be extended to 4 bytes. When transferring information to a BGP speaker that cannot handle 4-byte ASN numbers, the path attributes AS4_PATH and AS4_AGGREGATOR are included to include ASN information that cannot be contained in two bytes. The 4-byte ASN that cannot be contained in 2-byte ASN are replaced by AS_TRANS (AS23456) in the AS_PATH attribute. AS_TRANS is a reserved ASN which indicates that the original ASN was a 4-byte ASN that was non-mappable to a 2-byte ASN.

2.3. Routing Policies

Because network operators are often commercial companies, it is often undesirable to allow all traffic to flow through their AS. Larger network operators charge smaller network operators for gaining access to the global Internet by connecting to them, and they pay even larger network operators for complete connection to the Internet. The traffic that flows between ASes is regulated by routing policies. A *routing policy* is a policy that specifies which routes should be accepted from peers and which routes should be forwarded to other peers. BGP Routers can be configured extensively giving the administrator control over which route is propagated to which peer. By restricting the propagation of routes to peers it can prevent traffic from flowing through its own network.

2.3.1. Tiers

Network operators can be characterized as Tier 1, Tier 2, or Tier 3. A Tier 1 is characterized by not having any providers while still being fully connected to the Internet. In other words, it never has to pay anyone else for sending traffic to another AS. Tier 2 are larger network operator that pay Tier 1 networks for full connectivity to the Internet and have as their customers Tier 3 network operators. Examples of Tier 1 network operators are CenturyLink (formerly Level3), AT&T, NTT Communications, and KPN International. Tier 3 network operators are often smaller and provide Internet access to end-users.

2.3.2. Classification of Routing Policies

Routing policies can be very complex but can generally be characterized as either Customer to Provider (C2P), Provider to Customer (P2C), Peer to Peer (P2P), Tier 1 to Tier 1 (T2T) and Sibling to Sibling (S2S). This section will explain these routing policy types in more detail.

Customer to Provider

Two peers are said to have a *Customer to Provider (C2P)* relation if the customer pays the provider for transit to the global Internet. The provider then sends all routes it knows about to the customer such that the customer can reach other networks on the Internet. The provider also accepts the prefixes owned by the customer and propagates them to its other peers such that the prefixes of the customer are also reachable from other networks connected to the provider. The inverse of a customer to provider relation is a provider to customer (P2C) relation. A routing policy between the provider (AS1) and customer (AS2) can then be expressed using Routing Policy Specification Language [28] as is being shown in Listing 2.1. RPSL will be covered in more detail in Section 4.6, but for now it should suffice to know that a 'FROM' and 'TO' keywords followed by an AS express to which peer the routing policy applies and whether its a routing policy for incoming or outgoing traffic. The AS after the 'ANNOUNCE' and 'ACCEPT' keywords express which AS's prefixes are accepted. Note that AS2::CUSTOMERS represents all prefixes announced by the customers of AS2 and the prefixes owned by AS2 itself.

```
Routing Policies of AS1 (provider):
FROM AS2 ACCEPT AS2::CUSTOMERS
TO AS2 ANNOUNCE ANY

Routing Policies of AS2 (customer):
FROM AS1 ACCEPT ANY
TO AS1 ANNOUNCE AS2::CUSTOMERS
```

Figure 2.1: Routing Policy - Customer to Provider

Peer to Peer

One can speak of a *Peer to Peer (P2P)* relation when two ASes agree on exchanging traffic for their customers. Normally neither of them pays the other. Both ASes then send their own routes (including customer routes) to the other AS and forward the received routes to its customers only. It is important to note that the received routes from a peer should only be propagated to its customers and not to other peers or providers. The peer would otherwise be providing transit for traffic that is not originating from its own customers, which is not beneficial for the network operator. A Peer to Peer relation often saves money, since part of the traffic that

would normally be charged by a provider will now travel over the less-costly link with the peer. A schematic overview of routing policies is described in Listing 2.2.

```
Routing Policies of AS1:
FROM AS2 ACCEPT AS2::CUSTOMERS
TO AS2 ANNOUNCE AS2::CUSTOMERS

Routing Policies of AS2:
FROM AS1 ACCEPT AS1::CUSTOMERS
TO AS1 ANNOUNCE AS2::CUSTOMERS
```

Figure 2.2: Routing Policy - Peer to Peer

Tier 1 to Tier 1

When two Tier 1 ASes peer with each other they agree to freely exchange traffic. In this case, received routes are sent to all peers effectively causing the T1 AS to provide full transit to everyone that is connected. Customers pay for the traffic they send to Tier 1 ASes whereas other T1 providers are not charged for this. If they are charged for sending traffic they cannot be a T1 AS by definition. It is important to classify this separately from a P2P relationship since a Tier 1 A can propagate incoming routes from a Tier 1 B to another Tier 1 C which is not a customer of A. This would not be possible using a P2P relation. A schematic overview of routing policies as expressed in RPSL is shown in Listing 2.3:

```
Routing Policies of AS1:
FROM AS2 ACCEPT ANY
TO AS2 ANNOUNCE ANY

Routing Policies of AS2:
FROM AS1 ACCEPT ANY
TO AS1 ANNOUNCE ANY
```

Figure 2.3: Routing Policy - Tier 1 to Tier 1

Sibling to Sibling

There are multiple reasons why an organization might own multiple ASes. One reason is that they want to use different AS numbers in different areas of the world. Another reason is that a large organization owns multiple smaller companies of which some use a unique AS. When an organization uses multiple ASes they might decide to freely exchange traffic for the benefit of the larger organization. This would mean that they both exchange all routes, both providing transit for each other. It is often the case that AS X is providing transit for more traffic then the other AS Y is for AS X. But since free transit between ASes benefits the organization as a whole, this is not an issue. Note that Sibling to Sibling relationships are often much more complex than this and might not always exchange full traffic. Many reasons exist why all traffic is not exchanged. For example, some prefixes might have to be reachable with low latency which the sibling AS cannot provide. Or the sibling AS is not capable to handle the size of the traffic from the other sibling. Listing 2.4 shows a schematic overview of the routing policy belonging to a Sibling to Sibling relationship. Note that it is exactly the same routing policy as a Tier 1 to Tier 1 (T2T) relationship. S2S relationships can occur at smaller ASes as well, whereas a T2T relation only occurs at the largest ISPs.

```
Routing Policies of AS1:
FROM AS2 ACCEPT ANY
TO AS2 ANNOUNCE ANY

Routing Policies of AS2:
FROM AS1 ACCEPT ANY
TO AS1 ANNOUNCE ANY
```

Figure 2.4: Routing Policy - Sibling to Sibling

2.3.3. Internet Exchange Points

Instead of peering directly with other ASes, organizations often peer with each other by connecting to an *Internet Exchange Point*. An Internet Exchange Point (IX or IXP) is a physical infrastructure through which organizations can peer with each other or get transit. Because often many organizations connect at such an exchange point it is a cheap way to gain connectivity to many other networks. This is one of the key reasons why many ASes peer at one or multiple IXPs.

Route Servers

Internet Exchange Points often make use of *Route Servers* which are BGP speaking routers that forward BGP UPDATE messages to other peers that are participating at the Internet Exchange Point [44]. RFC7947 recommends route servers do this in a transparent way, making sure that the ASN associated with the route server is never visible to the peers, but some route servers do not because it is easier to configure. In this case, the ASN of the route server would turn up in the AS_PATH of BGP speakers elsewhere on the Internet. One reason that route servers do not include their ASN is that the AS_PATH of the route will get longer and might therefore not be picked as best route anymore.

2.4. BGP Anomalies

The Border Gateway Protocol was designed at a time where security was not a major concern. This is one of the reasons why the initial protocol contains a significant amount of vulnerabilities. Each BGP UPDATE can be modified by routers along the path and the protocol has no way of validating if an AS is authorized to announce a specific prefix. In this section, we will give an overview of common BGP anomalies occurring on the Internet. For more details on other types of BGP anomalies the reader is referred to [27].

2.4.1. Link Failure

The most simple type of BGP anomaly is that of a link failure. A *link failure* occurs when there are problems with the link between two routers. Because of this, messages cannot be delivered anymore and the peering session between the two BGP speakers will terminate. Other ASes might now be faced with an increased amount of traffic since the traffic cannot flow through this AS anymore and thus has to travel via another route. A link failure can have a variety of causes such as a defect physical connection or a misconfiguration of a BGP speaker which causes the inability to construct a peering session. Link failures are often due to an accident, but it is also possible that a malicious actor deliberately sabotaged the connection between two peers. A good example that caused link failure is the Slammer worm attack [47]. The Slammer worm caused an increased amount of traffic which some routers were not able to handle. Link failure occurred because routers were dropping BGP communication as well which in turn caused peering sessions to fail. This resulted in an increased amount of UPDATE messages as some BGP speakers kept re-establishing peering connections. Another example is the Mediterranean cable break in 2008 that caused disruption in more than 20 countries [31]. There are a variety of ways an attacker could perform a denial-of-service attack on a BGP speaker but those are outside the scope of this thesis and will not be covered.

2.4.2. Origin Hijack

Another common BGP anomaly is an *origin hijack*. In this case, an AS announces a prefix that it is not authorized to use. Often this generates a Multiple Origin AS (MOAS) announcement, where certain prefix is announced by more ASes, of which one is the offender AS. Since a more specific prefix is always preferred when routing traffic, accidentally or deliberately announcing a more specific prefix would cause all traffic to travel via the newly announced path. Depending on the properties of an announced route, such as AS_PATH length and whether its a more specific prefix or exactly the same prefix, part or all traffic is redirected to a new destination. Origin hijacks can be subdivided in either origin misconfiguration or intended hijacks. Because the effects of misconfiguration and intended hijacks are the same, it is often hard to distinguish them.

Origin Misconfiguration

When a prefix is accidentally announced by an AS that does not own or holds rights of the prefix, we speak of an *origin misconfiguration*. Most origin hijacks are due origin misconfiguration [54]. Since it is easy to make mistakes when configuring a BGP speaker, there is a high chance that a mistake is made when entering prefixes into the configuration of the BGP speaker. After discovering such an accidental announcement, the offender is contacted as soon as possible and the erroneous prefix is withdrawn. BGP speakers will now start using the previously used route (if any). A good example of an origin misconfiguration is that of the Pakistan Telecom incident [19]. In response to a censorship order from the government, a major ISP accidentally announced the prefixes used by YouTube causing many ASes to lose connectively to their service.

Intended Origin Hijack

Prefixes might also be hijacked by a malicious actor to perform Man-in-the-Middle attacks or to use the hijacked IP addresses for sending spam or other purposes. In this case, an attacker announces a prefix (or sub-prefix) from its own AS and optionally alters path attributes to make this new route the most preferred one. When successful, all or part of the traffic is sent to the attacker instead of the victim. It is often difficult to distinguish between an origin misconfiguration and an intended hijack since the visible behavior is the same. Network operators that are suspected of intended origin hijacking often claim that it was just a misconfiguration. However, sometimes action is taken by stopping contracts between such an AS and a provider, effectively shutting it down from the global Internet [8].

2.4.3. Path Hijack

Every AS on a path from origin to destination is capable of altering any path attribute on its way. A *path hijack* occurs when an AS modifies the AS_PATH in such a way that it appears to be connected to an AS which lies closer to the origin AS. An example can be seen in Figure 2.5 where AS5 fakes a connection to AS4 by sending an AS_PATH of AS5, AS4 to AS1.

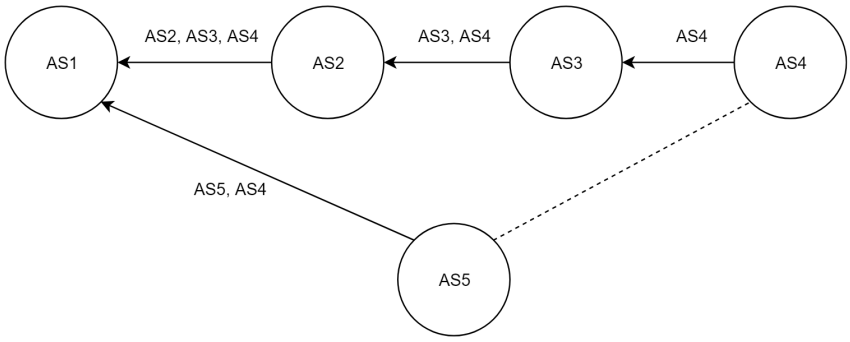


Figure 2.5: Example of a Path Hijack

This way traffic that would normally be sent via another route might now be sent to the malicious AS because the AS_PATH is shorter than that of other routes. An AS that performs a path hijack can then perform all kinds of man-in-the-middle attacks, such as snooping on traffic, redirecting traffic, etc. Because for an outsider it is often not clear which ASes are connected to each other (new connections may be formed and old ones may seize to exist) it is hard to detect path hijacks. Whereas origin hijacks can be easily triggered due a misconfiguration, with path hijacks this is not the case.

2.4.4. Route Leakage

According to RFC7908 [58] a *route leak* is defined as follows: "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. ..." An example of such a route leak can be seen in Figure 2.6. AS1 has a Peer to Peer (P2P) relationship with AS2 and therefore agrees to only exchange traffic between customers. However if the announcements from AS1 are accidentally sent to AS3 as well, traffic from AS1 destined to the rest of the Internet might now travel through and AS2 and AS3. Because AS2 is faced with larger amounts of traffic and does not benefit from this, such behavior is often not desired and is often a result of a misconfiguration of routing policies. We call AS2 the offender in this scenario since it should not have propagated the announcement to its provider. In this section, we will use *'peer'* to denote peers with which a Peer to Peer (P2P) relationship is formed unless stated otherwise.

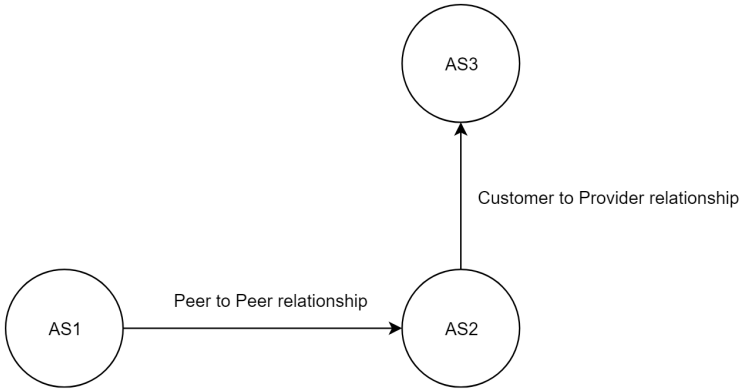


Figure 2.6: Example of a Route leak

Route leaks are often due to export misconfiguration where the intended policies are not correctly configured on the BGP speaker. However, it is also possible that a party causes a route leak intentionally to redirect traffic through their AS. Though, just as with origin hijacks, an attacker could pretend that it was just a misconfiguration. Since traffic will often flow differently after the route leak, the offending AS that leaked the route often is faced with a larger volume of traffic for the leaked prefixes. If the increased volume of traffic cannot be handled packets are often dropped on their way to their destination. A larger amount of dropped packets may, therefore, indicate a route leak. It is often difficult to detect route leaks because only the network operator knows the intended policies of its AS. Often smaller prefixes that would normally be aggregated, might become visible on the Internet as well. BGPSec Path Validation and Origin Validation using RPKI do not offer any protection against route leaks as is described in [59]. Route leaks can be further classified in multiple categories [58].

Provider to Customer to Provider leak This type of route leakage occurs when an AS received an announcement from a provider and propagates this to another provider as well. Because BGP speakers are often configured to ASes prefer customer routes over routes received by a peer or provider, a leaked route of this type often get preference over other routes. The traffic might still reach its destination unless the offending AS cannot handle the amount of traffic that is now re-routed through him. An example of this is the Dodo-Telstra incident where Dodo accidentally sent all their routes to its provider Telstra [13]. Because Dodo is a customer of Telstra, these routes were handled with preference over other routes. This resulted in an increased amount of traffic that had to travel through Dodo's and Telstra's ASes. Because of this a large amount of traffic of Dodo's providers now traveled via Dodo which resulted in major disruptions.

Peer to Peer to Peer leak This type of route leakage occurs when an AS sends the routes received by one peer to another peer. Normally only customer prefixes and the AS's own prefixes are announced to peers. When an AS announces the customer prefixes of one peer to another peer, it will effectively provide transit. This is undesirable because the leaking AS would be handling traffic that neither originates or is destined for its customers. [15] lists an incident where AS3561 leaked a large number of routes received from peers to other peers, effectively providing unintended transit for other ASes.

Provider to Customer to Peer leak Route leaks can also be caused when routes of a provider are leaked to peers. Routes received by peers are often preferred over other routes since they are less costly than sending traffic via a provider. If such a route leak occurs, traffic for each leaked route is likely to be sent to the offending peer, instead of the provider. [15] showed that some incidents leaked their routes in this way.

Peer to Peer to Provider leak Another type of route leaks occurs when a route received from a peer is sent to its provider. Because customer routes are often preferred, traffic is likely to be sent via the leaking peer, and the AS will be providing transit for traffic destined to the peer. An example of such an incident is the Axcelx-Hibernia case [7] where the routes from Axcelx, including routes received from peers were leaked to its provider Hibernia. This resulted in major disruption for services like Amazon which were among the prefixes being leaked.

3

Related Work

The Border Gateway Protocol is an essential protocol for exchanging routing information on today's Internet. Much research has already been performed on preventing and detecting BGP anomalies. This chapter will cover prior work that has been done to either protect from origin hijacks or detecting them such that the impact of an attack can be reduced. First, we will discuss technologies that aim to prevent BGP origin hijacks from happening such as Route Origin Authorizations and BGPsec and show why they are, in their current state, not yet capable of protecting the BGP ecosystem against origin hijacks. Afterward, we will cover a variety of important works on detecting BGP anomalies from 2006 till 2018. Many of these approaches suffer from significant limitations such as large false positive rates, the inability to detect origin hijacks quickly, or are only able to detect a subset of the origin hijacks visible to an RRC.

Route Origin Authorization

A Route Origin Authorization (ROA) [50] is a statement that gives an AS permission to announce a prefix. ROAs are used to verify if an announced prefix is announced by a legitimate origin. A router can then be configured to behave differently, for example by giving the newly learned route a lower preference so that older routes will be still preferred over a potential hijacked one. Creating a ROA consists of a few steps. First, a public and private key pair have to be generated by the organization that owns the prefix. Afterward, the public key should be communicated to a Trust Anchor which is usually a RIR but can be another network operator as well. The private key should be kept secret by the organization that owns the prefix. Then a ROA can be constructed stating that a particular AS is allowed to advertise a prefix and potential sub-prefixes up to a specified maximum length. After the organization signs the ROA with their private key, it is sent to the Trust Anchor. If the ROA is signed correctly and the resource is determined to be owned by the creator, the ROA will be published in a ROA repository such that other parties can use this ROA to validate routes to this prefix. ROA repositories are usually managed by the Trusted Anchor.

In [64] an analysis was done on detecting origin hijacks using ROAs. At the time of writing in 2012, only 2% of the prefixes used on the Internet was covered by a ROA. 1.6% of these prefixes were marked as valid, whereas 0.4% was marked as invalid. Announcements marked as invalid were either due to having a different origin than was specified in the ROA (20% of the invalid cases), had a longer prefix size than allowed (70%) or both (10%). Since at that moment in time only a very small percentage of the Internet's prefixes was covered by a ROA, it was clear that ROAs could not yet be used for complete detection and prevention of BGP origin hijacks unless adoption would grow.

The RPKI Monitor created and maintained by NIST [16] generates daily statistics on the number of prefixes for which ROAs have been created. On the statistics of 18 December 2018, we can see that only 10.70% of the prefixes currently being used on the Internet is considered valid by ROA objects. 0.88% is marked as invalid by a ROA and 88.43% is not covered by any ROA at all. We can conclude from this that the adoption rate of ROAs only increased by roughly 8 percent in 6 years. Because of this slow adoption rate, still only a small part of the prefixes that can be verified using ROAs. Because origin hijacks of prefixes that are not covered by ROAs cannot be detected, ROAs are currently still unsuitable as a standalone solution for detecting origin hijacks.

A report by APNIC also stated that prior to October 2017 only 3 ASes were found to use ROAs to actively filter invalid routes [1]. The reason why so few network operators have incorporated validation using ROAs in their BGP speakers is due to multiple reasons. One reason is that a significant amount of processing power is needed to cryptographically verify the signatures of ROAs. Validation using ROAs also require the availability of ROA repositories. Once these repositories become unavailable, a BGP speaker might not be able to correctly verify announcements. Together with the fact that ROAs only cover roughly 11% of the Internet's prefixes resulted in a low incentive to deploy RPKI on BGP speakers.

BGPSEC

Even if ROAs would gain adoption, it is not able to detect all types of attacks. An attacker is still able to hijack traffic by simply sending a fake announcement that contains the authorized origin at the end of the AS_PATH. This would cause it to seem that the legitimate owner is announcing the prefix (and therefore will not be filtered). The hijacker however is now on the path to the victim and can freely manipulate the traffic as he desires. BGPsec [49] aims to solve this problem by validating an AS_PATH with cryptography. BGPsec is an extension to the BGP protocol and therefore needs to be implemented by BGP speakers in order to be used. BGPsec enabled routers replace the AS_PATH attribute with the BGPSEC_PATH attribute. The main difference between these attributes is that the BGPsec protocol also includes the ASN of the receiver and then signs it using a cryptographic signature. In order to protect the integrity of the signatures, BGPsec does not allow multiple prefixes to be sent in one UPDATE message. A separate UPDATE message has to be signed for each neighbor. BGPsec enabled routers that receive such a message can now verify the signatures at each step. For example, lets suppose we have a path AS1, AS2, AS3 and AS3 is the originator. A router receiving this announcement first verifies AS3 using a so-called Subject Key Identifier that refers to the RPKI certificate that was used to create a signature. The router also verifies that AS3 intended to send the announcement to AS2. It then verifies if AS2 also wanted to send the announcement to AS1. If so, the AS_PATH is regarded as valid.

There are however a significant amount of limitations to BGPsec. Because BGPsec includes cryptographic signatures for each AS it travels through and can only send one prefix per UPDATE, UPDATE messages grow in size and volume significantly. It was estimated by NIST that it increases the size of UPDATE messages by a factor of 15 [9]. BGP speakers would need more memory to store these signatures and also need more processing power to create and verify signatures upon receiving and forwarding requests. All of these reasons resulted in a rather slow adoption rate for BGPsec. Another reason why BGPsec adoption is staggering is that BGPsec only works when each AS along the path supports BGPsec. As long as there is at least one AS along the path that does not support BGPsec, the signatures will be dropped and BGPsec security cannot be provided. Because of this, BGPsec in its current deployment state is not yet able to protect the BGPsec ecosystem against path manipulation.

Detecting Hijacks

Because completely preventing origin hijacks is very hard due to the fact that all organizations should implement protective technologies, research has been shifted to detecting origin hijacks such that they can be quickly resolved in case they occur. These detection approaches can be categorized in detection techniques that use historical BGP data and techniques that use reachability checks. It should be noted that BGP anomaly detection using machine learning has also been done [51][35][53]. However, these researches only focus on detecting other types of BGP anomalies such as link failures and route leaks. Other approaches that use statistical pattern recognition such as [37] and [62] focus on route leaks and AS_PATH manipulation respectively and also do not cover origin hijacks. One of the latest researches on anomaly detection makes use of time series analysis, but only mentions the detection of route flapping [56].

Detection using Historical BGP Data

Pretty Good BGP [45] was one of the first researches to our knowledge that pointed out the importance of creating a time window in which network operators can mitigate origin hijacks by temporary blocking the propagation of a received route. It uses historical data to determine whether a route is trusted or not. A route is trusted if the prefix has been announced by the same origin before. If a new incoming route is not trusted, it is quarantined for an adjustable amount of days, called the *suspicious period*. While quarantined, the route will not be used during this period. If the route is still present in the RIB after the suspicious period has ended, it will be removed from the quarantine and may now be used. The benefits of taking this approach are that

potentially anomalous routes will not be used for the configured suspicious period. However, the significant downside is that untrusted but valid routes cannot be used for the full suspicious period duration. This can be a serious drawback if the suspicious period duration is set to a large interval. On the other hand, when the suspicious period duration is set to a smaller window, network operators have less time to respond to origin hijacks. It is also important to note that even though this approach does give other network operators a time window to resolve a potential problem in if a suspicious announcement is not mitigated within this time window, the origin hijack will be accepted as valid. Because the quarantine is very likely to contain many valid routes as well, the information inside the quarantine is unsuitable to alert other network operators with.

Because PGBGP does not attempt to detect origin hijacks but only aims to delay potential origin hijacks such that network operators have more time to respond to incidents, a system that network operators can use to detect origin hijacks with was still necessary. In 2006, the Prefix Hijack Alert System (PHAS) was introduced [48]. The PHAS system aimed to provide real-time notifications of potential origin hijacking. Users of this system need to register their prefixes and the origins that are allowed to announce their prefixes. Whenever a prefix is announced with an unregistered origin, the user receives a notification about this event. Users will also receive a notification when their prefixes are no longer announced by their registered AS for some amount of time. The notifications are sent by mail and using multiple mail servers in different prefix ranges was presented as a solution to avoid mails travelling over a hijacked path. This approach however focusses solely on detecting prefix hijacks of an organization's own prefixes. Origin hijacks of prefixes owned by partner organizations will not be detected and traffic originating from the organization deploying this system might still be redirected to a wrong origin. If the partner organization does not deploy protective measures and does not detect the origin hijack, the origin hijack might persist.

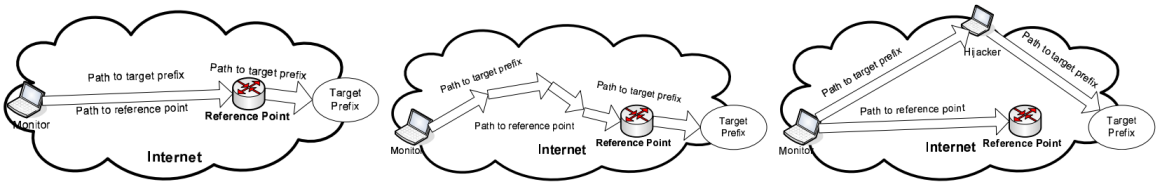
In 2009, NetReview improved on this by trying to detecting origin hijacks of other organizations as well [41]. The benefit of this is that, once origin hijacks against other organizations are detected, they can be informed of this and communication towards a hijacked prefix can be detected as well. NetReview aims to detect origin hijacks by comparing logs generated from BGP announcements against high-level specifications of expected routing behavior. These high-level specifications are created by network operators and shared with other network operators such that verification can be performed by all parties. NetReview also requires BGP speakers to maintain the logs for around one year. Although NetReview can successfully detect origin hijacks in real-time, network operators must share sensitive policy information with other network operators in order for the system to work. It also suffers from scalability problems caused by the amount of storage that log files require. In order to be effective, many network operators would have to share their policy information.

The Argus detection system, presented in [66], makes use of both traceroutes and BGP data and tries to establish a relation between them. This contrasts PGBGP, PHAS, and NetReview which only use BGP data. The key observation that is being built on, is that sending a probe from a router affected by an origin hijack will not receive a response and vice versa. Argus uses two months of historic BGP data to classify routes as normal or suspicious. It then tries to verify the suspicious routes by checking the reachability of these prefixes using traceroutes. The Internet Routing Registry (IRR) is also used to improve the false positive rate. Argus is able to detect origin hijacks in real-time but is unable to detect sub-prefix hijacking. It is the first paper to our knowledge that used routing registries to detect origin hijacks.

ARTEMIS [5] improves on the work of PHAS [48] by detecting more types of anomalies. The system is composed of three modules. The monitoring module monitors incoming BGP feeds from RIPE, Routeviews and optionally a local router. The detection module identifies hijacks by comparing monitor information with a specified configuration file which contains information about the ASes that are allowed to announce the prefixes owned by the organization that is being monitored. The mitigation module tries to recover from the attack once such an attack has been detected. ARTEMIS detects a variety of origin hijacks and even attempts to detect AS_PATH hijacking. ARTEMIS, however, inherits the drawbacks of PHAS and only focusses on detecting hijacks of the organization's own prefixes and does not attempt to detect origin hijacks of other prefixes. An organization deploying ARTEMIS is therefore still vulnerable against origin hijacks against unprotected organizations that they are communicating with.

Detection using Reachability Checks.

In 2007, [68] presented a way to detect BGP anomalies by using two observations. First the hop count of a path is generally stable and secondly, the path to a prefix is almost always a super-path of the path from the same source to a reference point on the previous path, as long as the reference point is close to the destination. The system uses vantage points at multiple locations and keeps track of hop counts to prefixes. If the amount of hops changes by an amount higher then the detection threshold a potential origin hijack may have occurred. If the path from the reference point is no longer a sub-path of the path to the destination, this is seen as an additional sign that an origin hijack may have occurred. To lower the number of false positives, each vantage point runs in parallel and the results are combined. It is however important that the placement of vantage points is critical to the success of this system. If the attacker is situated between the reference point and the destination, hijacks cannot be detected. Thus even though the paper claims to have both false positive and false negative ratios lower than 0.5 percent, this system is unable to detect all origin hijacks that it could observe. The system also suffers from major drawbacks. Namely, in order for the system to work and detect origin hijacks on multiple prefixes, many vantage points would have to be deployed. The system also requires more time in order to detect a hijack since the results of multiple vantage points need to be combined.



(a) Original (b) Legitimate route change (c) Prefix hijacking

Another approach was taken by [43], which is based on the observation that host-based and network-based fingerprints of endpoints in a hijacked network must be different than the original network. The system can be deployed inside a network owned by the organization and does not require external vantage points as was the case with [68]. The system sends probes to monitored prefixes in a cyclic manner. It does not require end-host cooperation since it uses a combination of traceroutes, ping and TCP-ping as probes. When an origin hijack occurs, the probes will not reach the correct destination anymore and will, therefore, be answered (if at all) differently. To avoid many false positives, the results of a previous cycle of probes are compared to the current cycle and if a specified amount of IP addresses is not reachable anymore, the alarm is triggered. The approach taken however does have some significant limitations. [43] states that even in the case of a MOAS, the destination is always a uniquely numbered network and that traffic will always arrive at the same physical network location. There are however cases, such as anycast, which allows a network operator to provide a service at physically distinct locations, potentially even showing different behavior per location. In this case, the system would erroneously report an origin hijack since the fingerprint at each anycast location is different. This solution also requires active scanning of networks which creates an often undesired load on the network. Multiple probing locations are necessary in order to cover origin hijacks that have only a limited impact. A sophisticated attacker is also able to fake the fingerprints of the original attacker, which enables him to go undetected.

[61] uses a similar approach as [43] but instead of running probe tests on a scheduled basis, it only probes when a new UPDATE message has been received. This would create less overhead on the network as prefixes that have not changed will not be probed any more. This approach requires the use of two or more ASes, that upon receiving an UPDATE message, all perform a ping test on the received prefix. Since some of the ASes might be affected by an origin hijack and some might not be, the results of the ping tests might be different in case of an origin hijack. This approach however still faces the same problems as [43] namely that when using anycast, the physical destination of each network is not always the same. It might also take some time before UPDATE messages have reached all the probing ASes and detection times might, therefore, be a lot slower. If the ASes are placed far away from each other, a large part of the Internet might have already been affected by the origin hijack before the detection algorithm reaches a conclusion. If ASes are placed closer to each other,

some hijacks might not be detected since the UPDATE messages never reach any of the probing ASes.

The authors of the ISpy system built further on this [67] by making the observation that hijacking is likely to cause unreachability. They state in their paper that in order for an origin hijack on an edge AS to be successful, it will always have to pollute one or more provider transit ASes first. However, since the time of writing in 2008, many organizations have established peer to peer relations with each other. When such an origin hijack would occur, these peers and their customers, would also be affected. They recorded 3000 transit ASes, whereas this number has since then significantly increased due to the growth of the Internet. The system also relies on ICMP echo replies which might not always be enabled by routers along the path to the destination. Link failures can also be a result of erroneous results since no probe reply will be returned in this case. The paper states that the time necessary in order to detect an origin hijack is in the order of minutes. Because of these drawbacks, this system cannot be effectively used.

One of the latest researches [11] at this moment took a different approach. By monitoring the Time To Live (TTL) value in normal traffic a subtype of prefix hijacking, known as the interception attack, can be detected. Upon performing an interception attack, the attacker first hijacks a prefix and then forwards all hijacked traffic to the intended destination. By doing this the attacker can eavesdrop or manipulate the data without denying service to the victim. When traffic is traveling via the hijacker to the victim, an increased amount of hops have to be taken. This is directly visible in a decrease in TTL values of the received packets. By observing a sudden drop in TTL values, these type of origin hijack can be detected. Because this approach only focusses on interception attacks, it cannot detect other origin hijacks such as origin misconfiguration. Drops in TTL might also have other reasons such as link failure which would result in an increased amount of false positives.

Our solution

In this Chapter we covered the state of the art protection and detection techniques related to BGP origin hijacks. As we have shown, the current techniques all have significant drawbacks that limit their usability. We improve on previous work by presenting a solution that does not suffer from these drawbacks.

Our solution uses a filter-based approach, where each announced prefix passes through several filters that each try to validate the prefix. What makes the filter-based approach unique compared to other approaches is that prefix announcements are treated as origin hijacks unless counter proof is given. This means that whenever an erroneous prefix announcement has reached the Remote Route Collector (RRC) on which we are performing detection, an origin hijack will always be detected. This contrasts many other solutions. PHAS [48] and ARTEMIS [5] are not able to detect origin hijacks other than hijacks of prefixes owned by the detecting organization, which means that origin hijacks against organizations they are communicating with cannot be detected. Argus [66], [61], [67] and [43] do not detect origin hijacks where the attacker is able to mimic responses to probe requests. [68] is unable to detect origin hijacks that occur between the destination and the vantage point and [11] cannot detect origin hijacks where the TTL value has not changed more then the detection threshold value.

In order for detection results to be useful, it is important that the results do not contain many false positives. If the output of a detection system would contain many false positives, it cannot be effectively used for automatic notifications or mitigation because maintainers would need to go through the time-consuming process of manually have to verify each case. Many solutions, especially those that rely on the reachability of a prefix [43][61][67][11] suffer from higher false positive rates because link failures also affect the reachability of a prefix. One would suspect that our solution, that treats every non-validated announcement as an origin hijack, would generate many false positives but the contrary is true. We combine many data sources that, when used by itself, are not able to validate all prefixes. However, when combining these sources together we can perform accurate detection. By using IRR, RSEF Listings, RPKI ROAs, and CAIDA's AS-relationship dataset, 99.9997% of the prefixes announced on 1 October 2018 could be validated.

Another important aspect of origin hijack detection systems is the amount of time it takes to detect an origin hijack. Detecting origin hijacks earlier in time enables the victim to respond and recover from the incident faster. This often results in a reduced impact in case such an origin hijack occurs, because fewer ASes are af-

fected. However, some state of the art techniques such as [68], [61], and [11] are unable to detect such hijacks in real time, often taking minutes to reach a conclusion. Our proposed system is able to detect origin hijacks in less than a second and can process announcements from multiple BGP sources in parallel.

Whereas other approaches such as [41] and [68] require new external infrastructures, such as vantage points or a shared routing policy database, our solution makes use of existing infrastructure such as routing registries and ROA repositories. The advantage of this is that, as the adoption of these technologies grows, our solution will be able to validate more prefixes as well, leading to more accuracy. Introducing new infrastructure often leads to slow adoption rates as well.

Many of the current solutions also rely on active probing to determine whether an origin hijack has occurred [43][61][67][11]. This leads to an increased amount of bandwidth being used for each prefix that needs to be monitored. Our system monitors announcements passively and does not actively probe prefixes, which results in less bandwidth being used that could otherwise be used for normal traffic.

	Proposed System	[48]	[41]	[66]	[5]	[68]	[43]	[61]	[67]	[11]
Detects in realtime	V	V	V	V	V	X	V	X	V	V
Detects all origin hijacks	V	X	V	X	X	X	X	X	X	X
No external infrastructure	V	V	X	V	V	X	V	V	V	V
Low amount of false positives	V	V	V	V	V	V	X	X	X	X
No network overhead	V	V	V	X	V	X	X	X	X	V

Table 3.1: Comparison of each detection system

As can be seen in Table 3.1 our proposed system is the only approach that is not limited by the drawbacks present in other solutions. This makes our system an effective solution for detection origin hijacks.

4

Datasources

Because our detection system uses a variety of data sources to detect origin hijacks, it is important that we have solid knowledge of the strengths and weaknesses of each data source. In this chapter, the data sources used in our system will be explained. Section 4.1 will cover raw BGP control data obtained from RIPE RIS and Routeviews. Section 4.2 will cover the RIPEStat API which is used to retrieve the routing history of a prefix. Section 4.3 describes CAIDA's AS-relationship set one which services such as ASRank [6] are built. Section 4.4 covers the RIR Statistics Exchange Format (RSEF). It also explains how we used RSEF listings to generate a set of Sibling to Sibling relationships which CAIDA's AS-relationship set is not able to. Section 4.5 will cover Route Origin Authorization (ROA). This chapter ends with Section 4.6 which covers the Internet Routing Registry, and the non-IRR routing registries of ARIN and LACNIC. It will also give a short introduction to Routing Policy Specification Language (RPSL) which is used to describe data in IRR registries.

4.1. RIPE RIS and Oregon Routeviews

In order to detect origin hijacks, we need to obtain BGP data that we can work with. The BGP data used varies between RIB dumps containing a complete dump of all routes received from peers and archives with BGP UPDATE messages that were received during a specific time span. Both of these archives are generated at specific time intervals. This data forms the core of our analysis. The RIBs and UPDATE messages both use the *Multi-threaded Routing Toolkit (MRT)* [32] file format which is designed to carry routing information for protocols such as BGP, OSPF, and others.

RIPE Routing Information Service (RIS) provides a variety of *Remote Route Collectors (RRC)* [22]. An RRC is a BGP speaker that records all BGP messages it sees and makes them available for inspection. The peers of an RRC usually send all their routes to the RRC such that users of RRC data have more data to operate on. The RIPE RRCs normally do not generate any significant amount of traffic and do therefore not impact the peers they are peering with. The RIPE RRCs create a dump of BGP UPDATE messages every 5 minutes and a RIB dump every 8 hours. Currently, there are 23 RIPE RRCs on a variety of locations (usually connected to Internet Exchange Points) around the world. The earliest RRCs have been collecting data since 2001, and the latest RRC became operational in January 2018.

Just like RIPE RIS the Route Views Project from the University of Oregon also maintains RRCs that collect data for research purposes. They currently maintain 23 RRCs which create a dump of BGP UPDATE messages and RIBs on different moments in time (usually every 5 minutes and 2 hours respectively).

Since both of these data sources contain raw BGP data and contain the exact UPDATE messages that were received by peers, it forms the ground truth that we operate on. However, it should be noted that the BGPSEC path attribute is currently not included in any MRT archives since the archives would otherwise be more than 15 times as large [9]. Another interesting note is that not every peer connected to the RRCs always provide all routes inside their RIB table. This is likely due a peer treating the RRC as a P2P connection, which allows us to only see their customers prefixes. Why a peer would want to form a P2P relationship with an RRC remains unknown to us.

4.2. RIPEStat API

The RIPEStat API [24] is a web API that can be used to retrieve statistics and information derived from the BGP data collected by RIPE's RRCs. We use the API to collect a list of BGP announcements for a specific prefix in order to detect if an announcement has already been made earlier in time. The RIPEStat API is also used to compute for how long a suspected origin hijack has been considered alive. Because the result of the API calls is based on the raw BGP data collected by RIPE's RRCs, we can consider their outputs as trustworthy. An example JSON response that was obtained by querying the Routing History API with 193.0.0.0/21 can be seen in Figure 4.1.


```
{
  "origin": "2116",
  "prefixes": [
  {
    "timelines": [
    {
      "endtime": "2016-08-14T23:59:59",
      "full_peers_seeing": 3.0,
      "starttime": "2016-08-03T00:00:00"
    }, ...],
    "prefix": "193.0.0.0/16"
  }]
},
{
  "origin": "3333",
  "prefixes": [
  {
    "timelines": [
    {
      "endtime": "2000-09-07T23:59:59",
      "full_peers_seeing": 21.75,
      "starttime": "2000-08-15T00:00:00"
    }],
    "prefix": "193.0.0.0/22"
  }, ...],
}, ...
```


Figure 4.1: Example JSON response of the RIPEStat Routing History API

4.3. CAIDA AS relationships

CAIDA's AS relationship dataset contains a set of inferred relationships between ASes on the Internet [12]. The relationships between ASes are expressed as Customer to Provider (C2P), Provider to Customer (P2C). Peer to Peer (P2P) or Sibling to Sibling (S2S). The AS relationship dataset is generated using the algorithm described in [52]. Many papers have been written about inferring relationships between ASes on the Internet and all agree that inferring AS relationships is a very complex problem [38][39][40]. Some of the factors that make inferring these relationships difficult are:

1. Internet Exchange Points (IXPs) including their ASN in the AS_PATH when forwarding announcements to customers of the IXP. Even though the best current practice for IXPs using Route Servers is to not include their ASN in the AS_PATH [44], some IXPs still do this. There currently is no publicly known list of IXPs that show this behavior known to us. Because some IXPs include their ASN, inferring algorithms often are not able to classify the right relationships between customers of the IXP.

2. Some ASes use *AS-prepending*, adding their AS multiple times to the AS_PATH, to decrease the likelihood of a route being selected as the most preferred route. This can result in erroneous relationships in naive algorithms.

3. Some ASes operate on multiple physical locations using the same ASN. This makes it hard to determine the correct location of an AS when inspecting an AS_PATH. The routing policies of BGP speakers are often different per region. An organization might be a customer of an AS in America whereas it might be the provider of the same AS in China.

4. Routing policies are often complex and not expressible in C2P, P2C, P2P or S2S. Some organizations are non-profit and use routing policies that do not make sense from a business perspective. Their routing policies often cannot be classified as C2P, P2C or P2P.

5. The current state of the Internet contains route leaks. Because of this, inferring relationships using BGP data of the current routing state of the Internet might result erroneously inferred links.

6. ASes sometimes *re-originate* an announcement, removing the ASes that appeared previously in their AS_PATH. This results in announcements that look like they have been originated from the aggregator AS. Because customers of ASes that re-originate announcements are often not visible by RRC's, an AS might be seen as a small network operator, whereas it might actually be much larger. Algorithms that infer relationships based on the size of a network operator in terms of customers might produce wrong results because of this.

7. BGP community values [33] do not follow a standard format. Each ISP uses different values for different purposes and might change the meaning of community values at any moment in time. Therefore BGP community values are not very reliable when being used for inferring AS relationships.

8. Mapping IP addresses to AS numbers introduce some error. Mapping an IP address to an ASN is often done by looking at routing registries or by using the AS that is currently announcing the IP prefix. Both of these are not guaranteed to be error-free because of origin hijacks or outdated route objects. This makes it harder to compare traceroutes with raw BGP data. There is no complete authoritative source that lists which prefix is owned by what AS. Wrongly inferred links can have a negative impact on the links that are computed at a later stage of inference algorithms.

9. Sibling to Sibling relations are often very hard to detect. We have included a way to infer Sibling to Sibling links using RSEF in Section 4.4.1. However, some ASes listed in RSEF listings are registered by a different entity, whereas they do actually belong to the same organization. An example of this the USA Information Systems Command (USAISC) which used to be a body of the Department of the Army but is now dismantled. There are however still prefixes registered under the USAISC, while in reality they now belong to the US Army Signal Command (USASC).

Because of these reasons the AS relationship dataset sometimes contains erroneous AS relationships, especially at higher levels of the Internet where routing policies often deviate from the common C2P, P2C, and P2P relationships.

4.4. RIR Statistics Exchange Format

Each Regional Internet Registry (RIR) publishes a daily listing of all the IPv4, IPv6 and ASN numbers that they have assigned. These listings are formatted using the RIR Statistics Exchange Format (RSEF) [23]. The Local Internet Registries (LIRs) and National Internet Registries (NIRs) that are assigned such prefixes then allocate sub-prefixes of these prefixes to customers. Only the assignments made by RIRs are present in their RSEF listing. Subsequent allocations done by LIRs or NIRs are not listed here. Each entry listed in an RSEF listing consists of the fields described in Listing 4.2.

```
registry|cc|type|start|value|date|status[|extensions...]
```

Figure 4.2: RIR Statistics Exchange Format Entry

The 'registry' field contains the RIR that generated this entry, 'cc' is the country code belonging to the Internet resource. The 'type' field specifies the resource type and can be either IPv4, IPv6 or an ASN. The 'value' field determines the prefix size or the amount of ASN's in this assigned block. The moment of registration

if recorded in the 'date' property and 'status' tells us the status of resource in question, such as reserved or assigned.

An extended version of the RSEF listing is also published by each RIR. In the extended RSEF listings a RIR is free to add additional information at the end of each record. In practice, however, only one extra field, called opaque-id, is added to the end of each entry. The opaque-id field represents the owner of the Internet resource. All records in the extended RSEF file with the same opaque-id are owned by the same resource holder [23]. An example of each type of resource can be seen in Listing 4.3.

```
ripencc|ES|asn|206252|1|20170227|allocated|55b2bf17-c6c4-451f-8aa7-d1fc04b295a4
ripencc|ES|ipv4|195.53.0.0|65536|19961218|allocated|55b2bf17-c6c4-451f-8aa7-d1fc04b295a4
ripencc|ES|ipv6|2a02:9000::|23|20110302|allocated|55b2bf17-c6c4-451f-8aa7-d1fc04b295a4
```

Figure 4.3: RIR Statistics Exchange Format Entry

As can be seen in Listing 4.3 all of these resources belong to the same organization because the opaque-id appended at the end of each entry is equal. Note that the value field for IPv4 resources specifies the number of addresses assigned. The value 65536 represents a /16 prefix because 2^{16} allows for 65536 unique IPv4 addresses) For IPv6 records the value field specifies the prefix length.

The information provided in the extended RSEF listings provide extra insights into the relation between ASes. Some large organizations often hold multiple AS numbers. When looked at from a network perspective, we can consider the relationships between ASes that belong to the same umbrella organization as Sibling to Sibling relationships since they share a mutual larger goal. It also often occurs that prefixes that are registered with a particular AS are later announced by another AS which is also owned by the same resource holder. These listings serve an authoritative source because they are being maintained by each RIR themselves. They are therefore unlikely to contain any errors.

4.4.1. Using RSEF Listings to discover S2S relationships

Because CAIDA's AS-relationship dataset does not include Sibling to Sibling relationships (S2S), an attempt was made to discover S2S relationships using RSEF Listings. In order to find ASes that belong to the same organization (and thus form a S2S relationship) all ASes with an identical opaque-id were grouped together. Using this approach we were able to create 6032 sets of ASes that contain the same opaque-id using the RSEF listings on 1 October 2018. Each set contains at a minimum 2 ASes, because no relationships can be inferred if an opaque-id applies to only one AS. It is important to note that not every AS in such a set is connected to each other and sometimes a large block of ASNs is assigned to an organization of which some are not yet in use. Therefore these relationships should only be used to override existing ones and not to append new relationships since physical connections between these ASes might not be present at all. When overriding CAIDA's AS-relationship dataset we found that 6337 links could be converted to S2S links.

4.5. RPKI - Route Origin Authorization

As has been explained in Chapter 3, ROAs provide additional security but are, in its current state, not able to validate all prefixes because of their low adoption. However, it is a great source for validating a part of the prefixes that are used on the Internet. Since ROAs can only be created by resource holders that own the correct private keys, we can assume that this data is correct. However, it is possible that a ROA is outdated or that the creator misconfigured its ROAs. Using the RPKI Validator [26] developed by RIPE we can see which announcements can be validated by their corresponding ROAs. Using the RPKI Validator, the creators of a ROA can be automatically mailed in case an invalid announcement is detected. In order to use ROAs in our detection algorithm we used to RPKI Validator to create a dump of all the ROAs on a daily basis. This enabled us to validate prefixes without having to send network requests for each prefix.

Each record in the ROA dump follows a standard format. An example can be seen in Figure 4.4. The 'prefix' and 'asn' fields specify which AS is allowed to announce what prefix. In case a network operator also wants to allow an AS to announce sub-prefixes of this prefix, the 'maxLength' field can be set to indicate the

smallest sub-prefix that may be announced. The 'ta' field specifies the Trust Anchor which signed the ROA.ust Anchor which signed the ROA.

```
{
"asn" : "AS42",
"prefix" : "74.63.16.0/20",
"maxLength" : 24,
"ta" : "ARIN"
}
```

Figure 4.4: RIR Statistics Exchange Format Entry

4.6. The Internet Routing Registry

The Internet Routing Registry (IRR) [14] is the union of worldwide routing policy databases that use the Routing Policy Specification Language (RPSL) [28]. These routing policy databases, which are often called *routing registries*, contain data about which AS is allowed to announce a prefix and which routing policies are applied on each AS. Often these routing registries are part of a larger database that also contains an address registry. For example, the RIPE Internet Number Registry (address registry) and the RIPE Internet Routing Registry (routing registry) are part of the same logical database. *Address registries*, maintained by RIRs, contain information about assigned Internet resources such as ASN, IPv4 prefixes, and IPv6 prefixes. Each of the five Regional Internet Registries (ARIN, APNIC, RIPE, AFRINIC and LACNIC) has their own address registry that can be accessed by the WHOIS protocol [34]. All RIRs, except for LACNIC, also maintains their own routing registry. Other large routing registries include LEVEL3, RADb, and REACH. Data inside address registries such as the allocation status of a prefix is provided by the associated RIR whereas information inside routing registries, such as routing policies and route information, is recorded by the assigned maintainers.

4.6.1. LACNIC and ARIN

Even though ARIN is part of the IRR and does have a routing registry containing RPSL objects, it also has a separate routing registry using their own data format different from RPSL. This non-RPSL registry is not part of the IRR is used more often. Because of this the data inside ARIN's IRR registry is often not up to date and lacking data that is present in their native registry.

LACNIC is not part the IRR and does not have a routing registry. It does, however, provide a web interface with proxy access to the routing registries of NIC Brazil, NIC Chile, and NIC Mexico. This is however limited to 30 queries per 5 minutes. NIC Brazil, NIC Chile and NIC Mexico do not offer bulk access to their routing registries.

4.6.2. Routing Policy Specification Language

The Routing Policy Specification Language [28] is used to define the data inside routing and address registries. RPSL is a rework of the older RIPE181 language [29] which has been in use since 1995. All routing registries that are part of the IRR now use this language. RPSL is an object-oriented language each object consists of a number of lines that specify a property of an object. The first property listed specifies the type of the object. An example of an object with type *route* is shown in Listing 4.5.

```
aut-num: AS3356
mnt-by: LEVEL3-MNT
as-name: Level3
```

Figure 4.5: RPSL - Route Object Example

The above RPSL object specifies that the AS3356 is maintained by LEVEL3. RPSL is a rich language and provides many types of objects, including aut-num for describing an AS, mntner for describing a maintainer and inet-rtr objects for describing Internet routers. Since our detection algorithm will only use the route object, we will not cover other data types. For more in-depth knowledge on RPSL readers are referred to [28].

4.6.3. Route Objects

The RPSL route object (and its route6 equivalent for IPv6 addresses) defines the relationship between a prefix and the AS by which it may be originated. This is one of the core objects used to specify routing information. Below you can see the general structure of such an object.

```
Attribute     Value
route:        <address-prefix>
origin:       <as-number>

Example:
route:          134.128.1.0/24
origin:         AS1234
```

Figure 4.6: RPSL - Route Object Format

The primary key of this object consists of the route and origin property. The route property specifies the prefix in question and the origin property describes the AS by which this prefix may be advertised. The RPSL specification [28] does not clearly specify how prefixes that are announced by multiple ASes should be handled but according to the APNIC RPSL documentation, a network is multi-homed if multiple route objects exist for the same prefix with different origin attributes [2]. Route objects are sometimes used by larger ISPs to generate a routing policy for their routers [17]. Customers of the ISP are then forced to enter and maintain their route objects in a routing registry, or their prefix announcements might not be forwarded onto the Internet by its provider.

5

Detecting Origin Hijacks

This chapter will explain how the data sources, described earlier in Chapter 4, are used in our detection system to detect origin hijacks. Our detection system uses a filter-based approach. That is, each prefix contained in an UPDATE message is processed individually and passes through several filters. At each filter, a prefix is checked on its validity. If a prefix announcement is determined to be valid by a filter it is removed from the detection pipeline. All prefixes that could not be validated after being processed by the last filter in the detection pipeline are considered origin hijacks. Each stream of UPDATE messages should be accompanied by a RIB that is in the state prior to when the UPDATE stream was received. For example, if we want to parse UPDATE messages from RIPE RRC3 starting at 10 October 2018 00:00:00, we also need the RIB of RRC3 that holds the routing state on 10 October 2018 00:00:00. From that moment onwards the RIB is updated when processing each update message sequentially and loading new RIBs is not necessary. It is important to note that our detection system only detects new origin hijacks. We do not attempt to detect origin hijacks that are already present on the Internet as we assume that these origin hijacks could have been detected previously in time by our system.

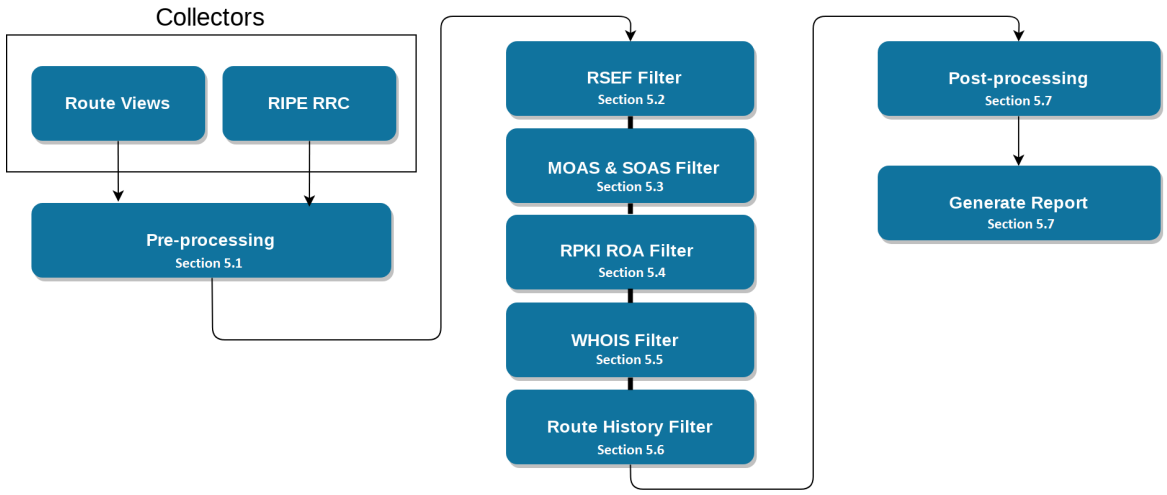


Figure 5.1: Overall Architecture of our Detection System

Figure 5.1 shows the overall architecture of our detection system. Each section in this chapter explains a step of our detection algorithm. Section 5.1 explains how we pre-process our data in order for it be usable by our detection algorithm. Section 5.2 explains how we use RSEF listings to validate prefixes by checking whether a prefix and its origin AS are both owned by the same owner. Section 5.3 will explain how our detection algorithm determines whether a MOAS or SOAS has occurred and how it uses this information to filter out announced prefixes that cannot be a new occurring hijack. Section 5.4 explains how we use RPKI Route

Origin Authorization (ROA) to filter out valid prefixes. Section 5.5 explains how we use routing registries to validate prefixes. Section 5.6 explains how we use RIPEStat's Routing History API to check if a prefix has been announced earlier in time by the same origin AS. This chapter concludes with Section 5.7 explaining how we post-process the prefix, now marked as a detected origin hijack, such that reports can be generated.

5.1. Pre-processing

Before we can process UPDATE messages we first need to download the required data sources. Afterward, the data sources are sanitized and transformed into a structure that can be efficiently used by our detection algorithm.

1. **Routing Information Bases**: RIBs are downloaded prior to scanning UPDATE messages. They hold the current routing state of the corresponding RRC that we are monitoring UPDATE messages for. RIBs are required in the MOAS and SOAS filter as will be described in Section 5.3. In order to reduce memory requirements, we only store the prefix and its associated AS_PATH and strip off all other path attributes. The AS_PATH attribute is stored in a compact form and is only parsed when it is accessed. A hashmap is used to speed up the lookup of prefixes significantly.

2. **UPDATE messages**: The UPDATE messages of the corresponding RRC are streamed upon demand. Once our detection algorithm has finished analysing the current UPDATE dump (which usually contains 5 minutes of UPDATE messages), it will automatically look for the next dump in time. Before UPDATE messages are processed they are first normalized such that subsequent actions performed by filters are less complex. All ASN fields are stored as 4-byte ASN numbers internally and MP_REACH_NLRI and MP_UNREACH_NLRI properties are converted to an internal structure that can handle IPv4 as well as IPv6 addresses. The properties that are converted to such an internal structure are AS4_AGGREGATOR, AS4_PATH, MP_REACH_NLRI, and MP_UNREACH_NLRI. No attempt was made to process BGPSEC_PATH path attributes since they are currently not included in any BGP control data sources that we know of. This is due to the increased amount of storage required when storing BGPSEC_PATH attributes.

3. **RSEF Listings** At midnight, prior to parsing UPDATE messages for that day, the RSEF listings for each RIR are downloaded. Because RSEF listings are small in size, each listing is parsed and stored completely in memory such that their entries can be quickly accessed.

4. **Routing Registries** Each routing registry is downloaded and sanitized at midnight. In order to have a usable dataset, we first remove all malformed IPv4, IPv6, and ASN entries. Because some routing registry dumps include historical route objects as well, we removed the duplicate prefix-origin pairs within a routing registry. Afterward, a binary tree of all route objects is constructed such that parent and sub-prefixes of a specific prefix can be quickly looked up. The binary tree is completely kept in memory to enhance performance.

5. **AS Relationships** CAIDA's AS Relationship dataset provides a monthly dump of the AS relationships on the Internet. Because these dumps are not made on a consistent basis, we use the most recent published version prior to the date we are processing UPDATEs messages for.

5.2. RIR Statistics Exchange Format Listings

The first filter makes use RIR Statistics Exchange Format (RSEF) listings to validate a prefix. As has been explained in Section 4.4, extended RSEF listings include an extra opaque-id field that represents the owner of the Internet resource. These opaque-id fields can be used to verify whether a prefix is being announced by its legitimate owner. This filter uses two methods to verify prefixes.

The first method looks for a prefix entry inside the RSEF listings that represents the prefix being validated by the filter. Because the prefixes found in an RSEF listing do not contain overlapping prefixes, we only have to look for the first entry that matches our prefix. If such a prefix entry is not found we cannot validate our prefix and the prefix moves on to the next method. When such a prefix entry is found, we also lookup RSEF entry corresponding to the origin AS announcing the prefix in question. If both of these are found we check if the RSEF entry associated with the origin has the same opaque-id as the RSEF entry associated with the inspected prefix. If this is the case, then the prefix has the same owner as the origin AS by which it was announced, and thus the originating organization exerts the right to announce this prefix. Prefixes that are

announced by an AS of the same owner are marked as valid and removed from the detection pipeline. As can be seen in Figure 5.2, upon receiving the prefix 138.255.124.0/24 originated by AS263973, we first lookup both entries and then compare the opaque-ids to infer that they have common owners.

```
Received Announcement: <138.255.124.0/24, AS263973>

1. Lookup the corresponding containing prefix entry:
lacnic|BR|ipv4|138.255.124.0|1024|20150901|allocated|240623

2. Lookup the corresponding ASN entry:
lacnic|BR|asn|263973|1|20150901|allocated|240623

3. Compare opaque-ids to identify a common owner.
The opaque-id of the prefix is 240623 and the opaque-id of
the ASN is also 240623, thus they have the same owner.
```

Figure 5.2: Approach 1: Using RSEF to identify a common owner

The second method to validate prefixes using RSEF is by finding a common owner between the origin stated in a matching route object and the origin inside the received UPDATE message. It often occurs that an organization owns multiple ASes and that prefixes are announced by multiple of these over time. If the origin of the announced prefix has the same opaque-id as the AS origin present in a route object that matches the prefix, then both belong to the same organization and the prefix is marked as valid. When a prefix is not marked as valid by either of these methods it continues to the next filter. As can be seen in Figure 5.3, first all associated route objects and the corresponding RSEF entries of their origin properties are retrieved. Then the RSEF entry of the origin inside the UPDATE message is looked up, and the opaque-id is compared to the opaque-id of the route object origins. If such a relation can be established, it means that the organization, that is allowed to announce the prefix, is just announcing the prefix with a different AS that is owned by them.

```
Received Announcement: <123.0.0.0/24, AS400>

1. Lookup corresponding route objects:
route: 123.0.0.0/24
origin: AS100

2. Lookup the origin entry of the route object:
rir|NL|asn|400|1|20150901|allocated|123456

3. Lookup the origin entry of the UPDATE message:
rir|NL|asn|100|1|20150901|allocated|123456

4. Compare opaque-ids to identify a common owner.
The opaque-id of the origin inside the route object is 12345 and
the opaque-id of RSEF entry belonging to the UPDATE origin is
also 12345. Thus the organization that was authorized to announce
the prefix as stated by the route object is the same organization
as the organization that is currently announcing the prefix,
only by a different AS.
```

Figure 5.3: Approach 2: Common owner between route object and origin

5.3. MOAS and SOAS

The second filter of our detection pipeline filters prefixes based on whether they introduce a Multiple Origin AS (MOAS) or a Single Origin AS (SOAS). The concept of MOAS and SOAS has been explained in Section 5.3. A SOAS occurs when a prefix is announced by a single origin whereas a MOAS occurs when a prefix is announced by multiple origins. In order to determine if a prefix introduces a MOAS or a SOAS, we compare the origin of the inspected announcement with the origins of routes for this prefix in the RIB. If the received origin is new and not already the origin of another existing route for the same prefix, we attach extra meta-information to this prefix stating that it introduces a new MOAS. When there are no other routes in the RIB for this prefix, we attach meta-information stating that it generates a SOAS.

Incoming prefix announcements that have the same origin as a route to this prefix that was already present in the RIB can never be the first signs of an origin hijack. In order for a route with the same origin to appear in the RIB another announcement had to be done previously. Because we are only trying to detect new occurrences of BGP hijacks we remove all prefixes that do not introduce a SOAS or a MOAS. If such an announcement was the result of an origin hijack, it would have been detected earlier in time. Since many announcements with origin changes are often already received from other peers, we can greatly reduce the number of prefixes that we have to validate in by only validating its first occurrence. Section 6.4 shows that 99.7699% of the prefixes can be filtered away using this filter when applied on the prefixes announced on 1 October 2018.

5.4. RPKI ROA

The RPKI ROA filter checks if there are any ROA objects that match the prefix-origin combination of the prefix under inspection. In order to speed up the validation process, we used RIPE's RPKI Validator [26] to create daily dumps of all ROAs and perform verification on these dumps. This way we do not have to perform any additional network requests, saving valuable time. A prefix can either generate a valid, invalid, non-existent match. When a prefix generates a valid match it is removed from the pipeline. If it is not marked as valid it continues to the next filter, regardless of whether it was invalid or non-existent. As has been explained in Section 4.5, ROAs are validated by a trusted third party and can, therefore, be considered as valid. Unlike route objects, the owner of an ROA is always verified. If the creator of an ROA does not own the prefix but tries to create an ROA for this prefix, it will not be accepted by the trusted third party. However it is possible that the origin specified in the ROA is misconfigured, but the chance that the misconfigured origin is the same as the ASN used by a potential hijacker is very low. An example where a prefix is validated can be seen in Figure 5.4.

```
Received Announcement: <212.94.32.0/19, AS28859>

1. Lookup the associated RPKI ROA:
AS: AS28859
Prefix: 212.94.32.0/19
Maximum Length: 19
Trusted Anchor: RIPE NCC RPKI Root

2. Compare the origin AS, and the listed ROA origin.
Because the ROA lists AS28859 as an authorized origin,
this prefix is considered valid.
```

Figure 5.4: Validating prefixes using ROAs

5.5. Routing Registry

The Routing Registry filter checks whether a matching route object exists for the prefix under inspection. Routing registries hold information on the routing policies of network operators. Route objects, contained in routing registries, describe which origin AS is allowed to announce a certain prefix. As will be shown in Section 6.1.2, roughly 92% of the routes inside a RIB have a corresponding route object. We will explain why each different type of match proves that the prefix announcement is valid since for some type of matches it is not always directly clear.

Exact Direct Match
With this type of match, the announced prefix matches with a route object that contains exactly the same prefix. The origin specified in the route object also directly matches the origin present in the announcement. This is the purest form of a route match since the route object directly expresses what is observed in the announcement. Roughly 62% of the prefixes inside a RIB generate this type of match. According to the RPSL specification [28], all prefixes should ideally generate this type of match. An example can be seen in Figure 5.5, where the prefix 1.2.3.0/24 has been announced by AS1 and the corresponding route object for 1.2.3.0/24 has its origin property set to AS1. The RRC sees the path AS2, AS1 and compares the prefix with existing route objects to verify that the prefix is indeed authorized to be announced by AS1.

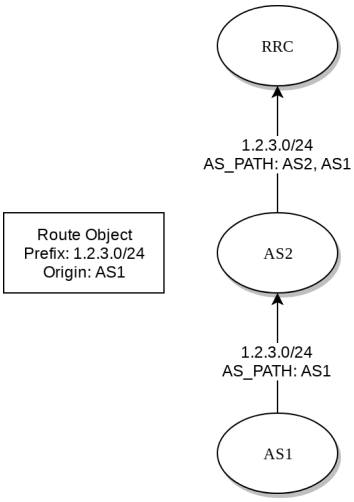


Figure 5.5: Exact Direct Match

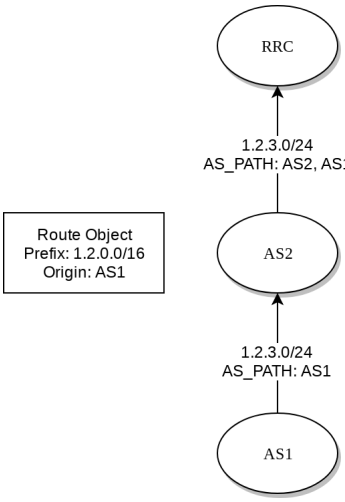


Figure 5.6: General Direct Match

General Direct Match
The announced prefix does not directly match with a route object of the exact same prefix, but matches with a route object that contains a larger prefix that encompasses this prefix. The origin inside the route object directly matches the origin of the announcement. Figure 5.6 shows an example where the prefix inside the UPDATE message is 1.2.3.0/24 with an AS_PATH ending with AS1. The matching route object for the same prefix is (1.2.0.0/16, AS1). It might be the case that, upon creating the route object, the creator had the intention that no sub-prefixes were allowed to be announced. But since the origins are matching, this would mean that the creator of the route object is itself violating its own policy. Therefore we consider this as a valid match since we do not regard an organization using its prefixes in an undesired way an origin hijack. 22% of all the prefixes inside a RIB generates this type of match.

Specific Direct Match
This type of match is generated when the announced prefix is matched against several route objects of more specific prefixes that have the same origin. This might be the result of a router aggregating multiple smaller prefixes into one larger prefix. Because the origin is exactly the same, unintended behavior could only come from the owner itself, and that is why it is considered a valid match. An example can be seen in Figure 5.7. The received prefix is 1.2.3.0/23 announced by the origin AS1. Two route objects were found: (1.2.3.0/24, AS1) and (1.2.4.0/24, AS1). Together these route objects cover the whole range that 123.0.0.0/23 covers. Only 0.89% of all prefixes inside a RIB generate this type of match.

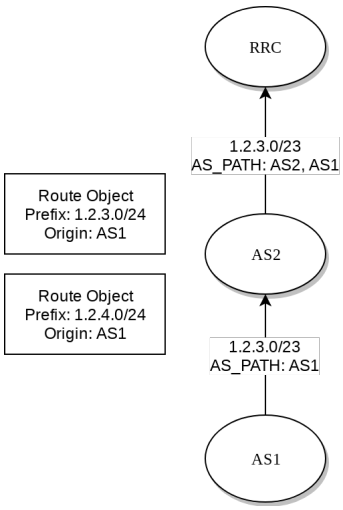


Figure 5.7: Specific Direct Match

Exact Upstream Match
This type of match occurs when a route object with the same prefix is found but the origin listed in the route object is an upstream of the origin of the announcement. This would mean that, in order for this origin hijack to succeed, the upstream accepted and forwarded the malicious UPDATE from one of its customers or peers and also prefers sending traffic to the hijacker instead of its own internal network. This would mean that the upstream, which is the owner of the prefixes, is also partly responsible for letting this origin hijack succeed. A much more plausible explanation for this scenario is that the upstream, which owns the prefix, registered a route object for itself but delegated its use to one of its customers. This is a common practice often seen on the Internet. For example, as shown in Figure 5.8, the received prefix is 1.2.3.0/24 and has an AS_PATH of AS2, AS1. The matching route object is (1.2.3.0/24, AS2). Because providers often register their prefixes on their customer's behalf we consider this a valid case. 1.29% of all the prefixes inside a RIB generate this type of match.

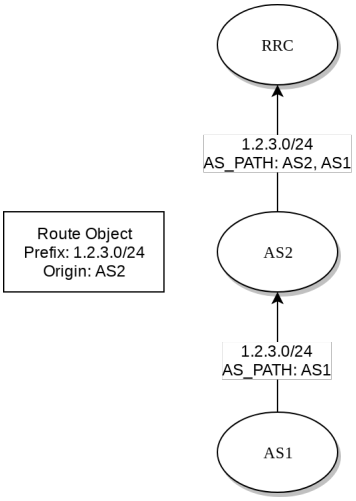


Figure 5.8: Exact Upstream Match

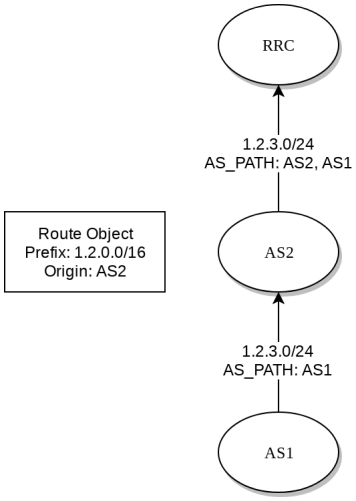


Figure 5.9: General Upstream Match

General Upstream Match
A General Upstream Match is similar to an Exact Upstream Match with the difference that the route object now contains a larger prefix that contains the announced prefix. This time the upstream registered its assigned prefix as a route object, and likely delegated a part of the prefix to a downstream customer. If this was a real origin hijack it could only succeed if the upstream forwarded the invalid announcement and redirected incoming traffic destined to himself to the attacker. An example can be seen in Figure 5.9. The received prefix is 1.2.3.0/24 and has an AS_PATH of AS2, AS1. The matching route object contains (1.2.0.0/16, AS2). Just as with an Exact Upstream Match, we consider this a valid case because in such a case the upstream could easily mitigate these kind of hijacks and an upstream provider registering its assigned (larger) prefix is much likelier. Less then 3% of all the prefixes inside a RIB generate this type of match.

Specific Upstream Match
In this case, the received prefix is matched against multiple route objects that are registered by an upstream. An example is shown in Figure 5.10. The received prefix is 1.2.3.0.0/23 and has an AS_PATH of AS2, AS1. The matching route objects contain (1.2.3.0/24, AS2) and (1.2.3.0/24, AS2) respectively. Just as with an Exact Upstream Match, the upstream could easily mitigate this by blocking these announcements and therefore it is likelier that the upstream registered multiple smaller prefix blocks of which multiple was assigned to the customer and then announced as an aggregate. Only 0.05% of the prefixes inside a RIB generate this type of match.

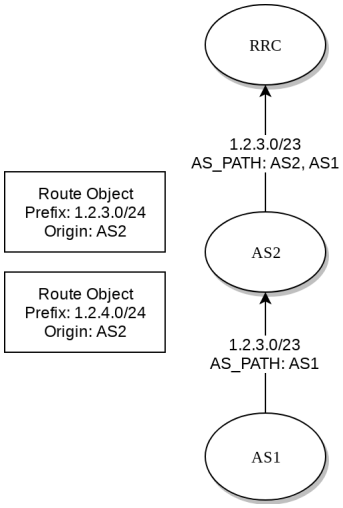


Figure 5.10: Specific Upstream Match

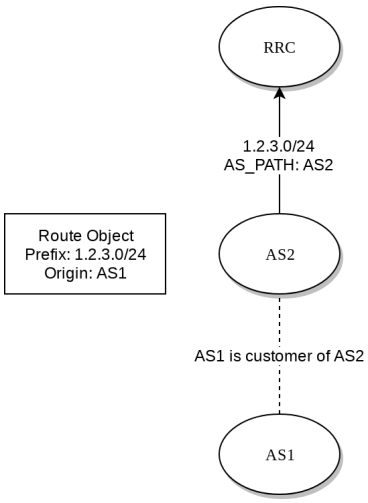


Figure 5.11: Exact Link Match

Exact Link Match

In this case, a route object with the exact same prefix was found and, according to the AS relationship dataset by CAIDA, the registered origin inside the route object is a customer of the origin in the announcement. For example, as shown in Figure 5.11, the received UPDATE contained a prefix 1.2.3.0/24 and the origin is AS2. The matched route object is (1.2.3.0/24, AS1). By checking if AS1 is a customer of AS2 we can check if such a match has occurred. This often occurs because upstream providers strip of the AS_PATH to their customers making it appear that they are the destination AS where traffic should travel to, while in reality the traffic is being forwarded to the customer. The process of stripping of the AS_PATH is also called re-origination since the origin is changed by this process. Since traffic always travels through the upstream, it would make no sense for the upstream to hijack this prefix by means of an origin hijack, since it already can manipulate and redirect traffic wherever it wants to. Therefore we assume this is a valid type of match. 0.98% of the prefixes inside a RIB generate this type of match.

General Link Match

Just as with an Exact Link Match, it would make no sense for an upstream to perform an origin hijack while it already has control of the traffic. An example can be seen in Figure 5.12. The received prefix is 1.2.3.0/24, its associated origin is AS2 and the matching route object is (1.2.0.0/16, AS1) where AS1 is a customer of AS2. The most likely scenario here is that the upstream is de-aggregating the prefixes in order to perform traffic engineering. The amount of prefixes inside a RIB that generate this type of match is 1.08%.

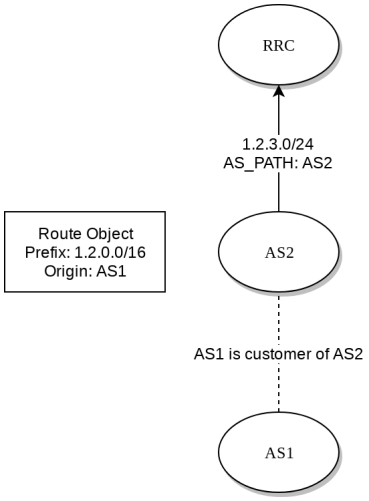


Figure 5.12: General Link Match

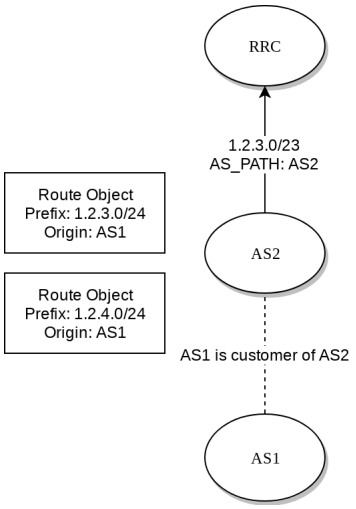


Figure 5.13: Specific Link Match

Specific Link Match

In this case, a received prefix is matched against several route objects that all have an origin of a customer of the origin associated with the received prefix. Just as with an Exact Link Match, it would give the upstream no extra benefits if it performed an origin hijack and this type of match is often seen due to the aggregation of prefixes. Figure 5.13 shows an example of this type of match. 1.2.3.0/23 was received with an origin of AS2. The associated route objects are (1.2.3.0/24, AS1) and (1.2.4.0/24, AS1) and AS2 is a customer of AS1. It often occurs that a provider aggregates these prefixes into a larger prefix. The path to the customers is then often removed the reduce the AS_PATH size. 0.29% of the prefixes inside a RIB generate this type of match.

Specific Mixed Match
This type of match is generated when a prefix is matched against multiple route objects that all generate a different type of Specific Matches. Because each type of Specific Match is on itself a valid case, the composite is considered a valid case as well. Only 0.03% of the prefixes inside a RIB generate this type of match. An example of a combined Specific Link Match and Specific Direct Match is shown in Figure 5.14.

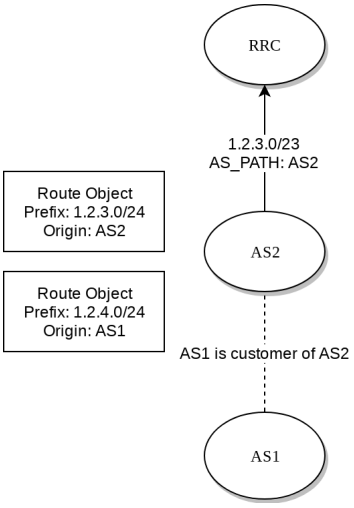


Figure 5.14: Specific Mixed Match

When a route object match is found the prefix is marked as valid and removed from the pipeline. Prefixes that do not have an associated route object match move on to the next filter. Section 6.4 will show that around 91% of the prefixes on 1 October 2018 could be validated by this filter. It is important to note that route objects are created manually by network operators themselves. Many routing registries require a membership before route objects can be created. Upon registration, the identity of a network operator is often verified to make sure that a new member is really who it claims to be [20][21][4]. Therefore, the chance of a malicious actor creating fake route objects is decreased. Even if an attacker is capable of creating fake route objects, it would be difficult for the authenticated member to explain why he created such route objects upon the discovery of such an origin hijack. Many routing registries also restrict users when trying to create route objects for resources they do not own. A misconfigured route object can only be exploited if the origin field would by chance contain the ASN of the hijacker. Thus we can assume with reasonable confidence that the information inside route objects is not malicious.

LACNIC
Whereas all routing registries provide a dump of their routing registries, LACNIC does not do so. The reason is that LACNIC does not own a routing registry but instead proxies requests to the routing registries of NIC Brazil, NIC Mexico, and NIC Chile. We asked each of these NICs if they could provide us with daily dumps of their registry but they were not able to do so. The maximum rate limit on the WHOIS services of NIC Brazil and NIC Mexico is 30 queries per 5 minutes which means we cannot retrieve all their prefixes on a daily basis. Therefore our implementation had to incorporate a separate filter that queries LACNIC for each prefix. We deliberately chose to put this as the last filter in our detection algorithm such that it has to be applied to as few prefixes as possible. This allows us to stay below the rate limit deployed by LACNIC, which is also 30 queries per 5 minutes and to perform as few network requests as possible.

5.6. Routing History
The Routing History filter, which is the last filter of our detection system, checks whether a prefix has been announced by the detected origin earlier in time. When the prefix, or a parent prefix, has already been announced by the same origin before, the hijack would have been detected earlier in time and we remove the prefix from the pipeline. Otherwise, it passes this filter. However, as a consequence of this filter, prefixes that were hijacked by the same hijacker previously in time, will not be detected anymore at a later moment in time. This problem can be easily solved if network operators using our system would confirm detected origin hijacks, such that, when a hijack by the same origin would occur again, it can be detected in the present time as well. Section 7.1.2 will discuss this problem in more detail. Prefixes that pass the entire pipeline will be exposed to additional analysis as described in Section 5.7.

Our detection system uses the RIPEStat Routing History API [24] to retrieve the announcement history of the prefix in question. We can look up the complete history for a prefix from 1999 onward (the moment the first RRC started running). It also retrieves the history of parent prefixes. It is important to note that the Routing History API is based on the RIB dumps that are made every 8 hours. This means that if a prefix was announced and withdrawn between the RIB dumps, its announcement would not be detected since it does not appear in the RIB at the moment of the dump. However, this does not pose a real problem since the aim of the filter is to filter away prefixes that have been announced by an origin for a longer time. If a prefix was announced for less then 8 hours between RIB dumps, it is still suspicious. Effectively this would mean that a prefix that would be marked as a hijack previously in time as well, will now be marked as a hijack once more because a prior announcement was not seen. The small number of duplicates that would be generated can be easily filtered out when desired.

5.7. Post-processing

For each prefix that is not filtered away during the detection algorithm, a report is generated containing the contents of the UPDATE message and extra meta information that was added by each filter in the detection pipeline. An example of such a report can be seen in Listing 5.15. We deliberately chose to use output reports in JSON format such that they can be easily consumed by other applications, for example, to visualize the reports.

```
{
        "introduces-soas": false,
        "introduces-moas": true,
        "same-resource-holder (prefix-origin)": false,
        "same-resource-holder (RO-origin)": false,
        "rpki-status": "nonexistent",
        "announced-earlier (exact)": false,
        "announced-earlier (parent)": false,
        "prefix": "208.75.144.0/24",
        "timestamp": "2018-09-20T06:41:30Z",
        "update-message": {
        "withdrawnRoutes": [],
        "pathattributes": {
                "ORIGIN": {...},
                "AS_PATH": {...},
                "NEXT_HOP": {...}
        },
        "nlri": [
                "208.75.145.0/24",
                "208.75.144.0/24"
        ]
}
```

Figure 5.15: An example hijack report in JSON format

Using the generated information, maintainers of the detection system can more easily analyze potential incidents. Note that the nlri property inside the update-message field contains the other prefixes that were contained in this UPDATE message. It might be the case that only some of the announced prefixes are invalid, whereas others are not. It is also often the case, as will be shown in Chapter 6, that multiple prefixes are hijacked in one UPDATE message. It is important to understand that it is hard to verify whether a certain prefix announcement is truly an origin hijack or not. Some network operators do not attempt to protect their prefixes by registering them at routing registries or by creating ROAs. Network operators might also temporary or permanently transfer prefixes to other organizations, which makes the RSEF filter unable to establish a relationship between the prefix owner and origin. Because the victim of an origin hijack loses connectivity in some or all parts of the worlds, BGP origin hijacks are often discovered and repaired within hours or days. We will analyze the duration of our detected origin hijacks in Chapter 6 and verify whether this is the case.

6

Results

This chapter contains the results that were obtained through our analysis. Section 6.1 assesses the accuracy of route objects within routing registries. Because the accuracy of route objects has been doubted for a long time, it is necessary to access its validity before it can be used with confidence in our detection system. Section 6.2 describes the results of running our origin hijack detection system from 20 September 2018 to 18 October 2018. In Section 6.3 we compare our detected origin hijacks with the origin hijacks detected by BGPStream in order to evaluate the performance of our detection system in terms of accuracy. Section 6.4 evaluates the performance in terms of speed for each filter in our detection system.

6.1. Accuracy of Routing Registries

The accuracy of routing registries has been a debate for many years [55][60]. So in order to detect origin hijacks using route objects, it should first be assessed that this information is reliable. We are only accessing the accuracy of route objects because our detection system only uses the route object data from routing registries to detect origin hijacks.

As has been described in Section 4.6.1, ARIN has their own non-IRR routing registry which contains more up-to-date information than their IRR registry. This routing registry does not contain routing policies but it contains data on which origin is allowed to announce a certain prefix. In order to incorporate this information in our analysis, we obtained ARIN bulk access and converted this information to prefix-origin pairs. LACNIC only provides bulk access to their address registry which does not contain route objects or routing policies as has been explained in Section 4.6.1. In order to obtain the prefix-origin pairs of the routing registries that LACNIC proxies we scraped these registries. This was done by setting up 10 different Virtual Machines with different IP addresses that each scrape a part of the address space within LACNIC's designated area, to bypass LACNIC's rate limit as much as possible. Because scraping each /24 prefix would be infeasible to do within a reasonable amount of time, we instead scraped each prefix inside LACNIC's RSEF listing. This resulted in a set of prefix-origin pairs that we could use in our analysis. We were able to scrape 15208 route objects using this approach, of which 9671 IPv4 prefixes and 5537 IPv6 prefixes. All IPv4 route objects together cover a space that is equal to 47% of the IPv4 address space designated to LACNIC. The amount of address space covered by IPv6 route objects of the IPv6 address space assigned to LACNIC is equal to 0.4%. This is largely due to the fact that many IPv6 prefixes are not allocated yet.

As has been discussed previously in Section 4.6.3 and Section 5.5 the primary key of a route object consists of the route property, which represents the prefix in question, and an origin property specifying the AS that may announce this prefix. Multiple route objects containing the same prefix but different origins indicate that a network is multi-homed or that a stale route object has not been removed. In order to check the validity of the existing route objects, we compare the route objects of each routing registry to the UPDATE messages and RIBs of RIPE and Route-views.

We split the analysis of route objects into two parts: First in Section 6.1.1 we verify how many route object within each routing registry can be matched against routes in the RIBs of RIPE and Routeviews. For each

routing registry the amount of route object matches, as has been previously explained in Section 5.5, will be listed. This is done to detect stale route objects inside routing registries that do not have a matching announced prefix anymore. Afterwards in Section 6.1.2 we will verify how many prefixes inside the RIBs and UPDATE messages have a matching routing object. This is done to see how many prefixes could be validated using routing registries. The results are obtained by using 43 RIBs of RIPE RRC and Routeviews RRC of 1 September 2018 00:00:00 and the corresponding routing registry snapshots of that day. Results using UPDATE messages are based on the UPDATE messages of RIPE RRC00 on 1 September 2018.

6.1.1. Analysing route objects from the Registry Perspective

In order to find potentially stale route objects, we compare the route objects of each routing registry to the RIBs of RIPE and Routeviews. Each RRC has a different view of the Internet, and some prefix-origin pairs might only be visible by some of them. Figure 6.1 shows the results for the routing registries of Regional Internet Registries. Figure 6.2 shows the results for large routing registries with more than 10000 route objects, whereas Figure 6.3 shows the results for small routing registries with less than 10000 route objects.

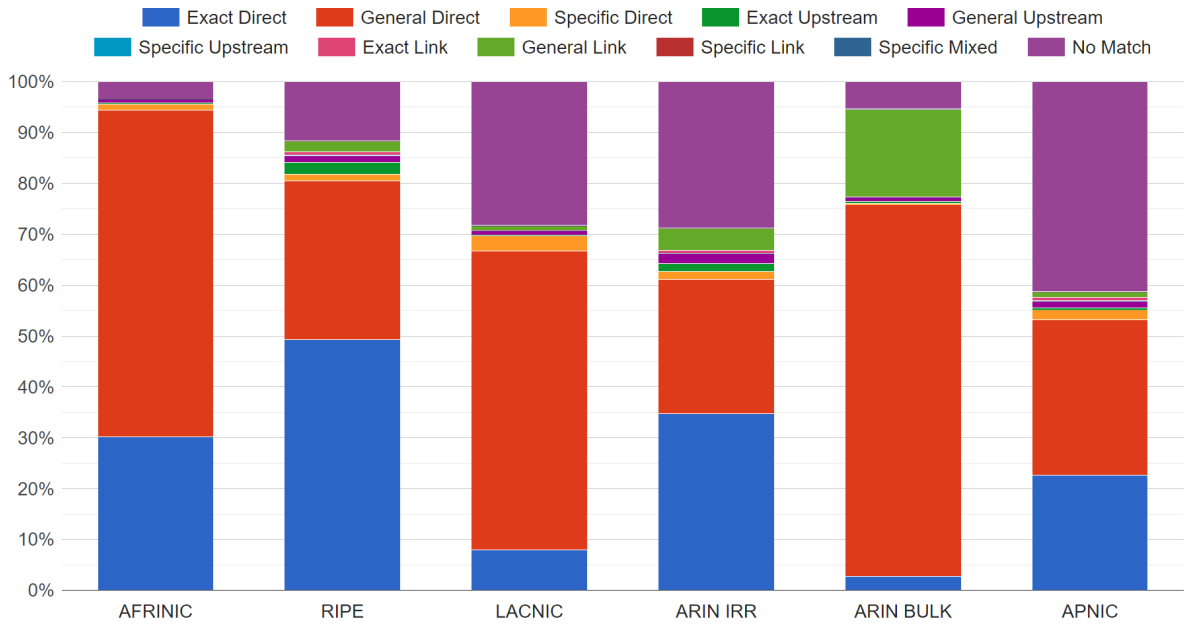


Figure 6.1: Route object matches for Regional Internet Registries

As can be seen in Figure 6.1 most route objects generate either an Exact Direct Match (where both the prefix and the origin have an exact match) or a General Direct Match (where a prefix in the RIB contains the prefix of the route object and both have the same origin). A large amount of General Direct Matches is due to the fact that a large number of network operators aggregate multiple smaller prefixes into a single larger prefix.

ARIN's non-IRR routing registry (as denoted by ARIN BULK) contains only a small percentage of route objects that do not generate a route object match. This non-IRR routing registry of ARIN also generates the largest percentage of General Direct Matches when compared to other RIRs. We suspect this is due to aggregation more often being performed within ARIN's designated area, compared to other areas in the world. The IRR registry of ARIN (as denoted by ARIN IRR) contains a high amount of route objects that do not generate a match. ARIN mainly focuses on their non-IRR routing registry. They have acknowledged, the increased amount of outdated information and created an IRR roadmap to set new goals to improve their IRR registry [3].

LACNIC contains the highest amount of route objects that do not generate a match. It should be noted that, because we were only able to scrape route objects for the prefixes inside LACNIC's RSEF listing, the results might be different from when we would have had access to all route objects. Because these results are obtained from route objects that cover 47% of LACNIC's designated IPv4 space, we suspect that these results would not significantly change if we had access to all route objects.

Compared to other RIRs AFRINIC's routing registry contains the smallest percentage of route objects that do not generate a match. This might be due the fact that AFRINIC was founded later in 2005 whereas the IRR was founded in 1995. This might have allowed AFRINIC to learn from the mistakes made in the past when operating a routing registry as well apply any established best practices. AFRINIC also shows a large amount of Generic Direct Matches compared to other RIRs indicating a increased amount of aggregation within AFRINIC's designated area of service.

RIPE contains the largest amount of Exact Direct matches and also has a relatively small amount of route objects that do not generate a match. As is shown in Table 6.1, the average age of each route object is generally high, which indicates that route objects are more often maintained.
When we compare APNIC to other RIRs, the percentage of route objects that generate no match is much larger. When inspecting the results on a deeper level we found that more than 50% of the route objects that generated no match had sub-prefixes that were being announced by the same origin. Because only part of the address space of the route object was announced and not the complete space a match was not generated. We were not able to identify why this was the case but we suspect that it is due to organizations only using part of their assigned address space but registering it as a whole.

Table 6.1 shows statistics on the number of duplicates and the average age of each routing registries. A route object is marked as a duplicate when a route for that prefix already exists. It should be noted that there are valid scenarios where multiple route objects exist such as anycast and MOAS configurations. The number of local duplicates refers to the number of duplicate routes within a routing registry, whereas the global duplicates take into account all the routes of each registry. For example, when a route to a certain prefix is present in both ARIN and AFRINIC it is marked as a duplicate in both registries. For the exact amount of matches per registry we refer the reader to Table A.2 in Appendix A.

Routing Registry	**Routes**	**Local Duplicates**	**Percentage**	**Average Age**	**Global Duplicates**	**Percentage**
ARIN BULK	3370804	4984	0.14%	Jan 09 2013	78562	2.33%
AFRINIC	19026	71	0.37%	Nov 05 2017	1986	10.43%
ALTDB	14783	920	6.22%	Apr 24 2012	8497	57.47%
AOLTW	153	12	7.84%	Mar 24 2008	102	66.66%
APNIC	204013	10341	5.06%	May 19 2013	42610	20.88%
ARIN	51851	2516	4.85%	Feb 19 2015	19994	38.56%
BBOI	3376	46	1.36%	Jul 21 2015	3179	94.16%
BELL	29472	160	0.54%	Dec 10 2000	10679	36.23%
CANARIE	1434	75	5.23%	Jun 08 2013	907	63.24%
EASYNET	38	0	0%	Nov 04 2008	19	50%
GW	4	0	0%	Dec 01 2000	3	75%
HOST	29	0	0%	Nov 24 2007	23	79.31%
JPIRR	10286	183	1.77%	Apr 25 2018	7293	70.90%
LEVEL3	127564	23989	18.80%	Jul 28 2007	74114	58.09%
NESTEGG	4	0	0%	Jul 06 2006	3	75%
NTTCOM	386267	51593	13.35%	Aug 16 2014	270765	70.09%
OPENFACE	19	0	0%	May 01 2008	17	89.47%
OTTIX	47	2	4.25%	Apr 21 2009	34	72.34%
PANIX	40	0	0%	Jan 13 2006	21	52.5%
RADB	979692	131287	13.40%	Oct 27 2013	403581	41.19%
REACH	28692	3379	11.77%	Apr 23 2010	23025	80.24%
RGNET	64	29	45.31%	Jul 28 2012	24	37.5%
RIPE	294209	18597	6.32%	Feb 04 2014	20244	6.88%
RISQ	903	17	1.88%	Sep 18 2005	785	86.93%
TC	6122	28	0.45%	Jul 18 2014	3549	57.97%

Table 6.1: Amount of duplicate route objects on 10-12-2018

Table 6.1 shows us that the routing registries of Regional Internet Registries generally have less local and global duplicates then other routing registries. ARIN's non-IRR registry has the least amount of local and global duplicates compared to other routing registries of RIRs. Compared to other routing registries, routing

registries of RIRs seem to have a higher average age and fewer duplicates. These are indicators that route objects within these registries are generally more up-to-date.

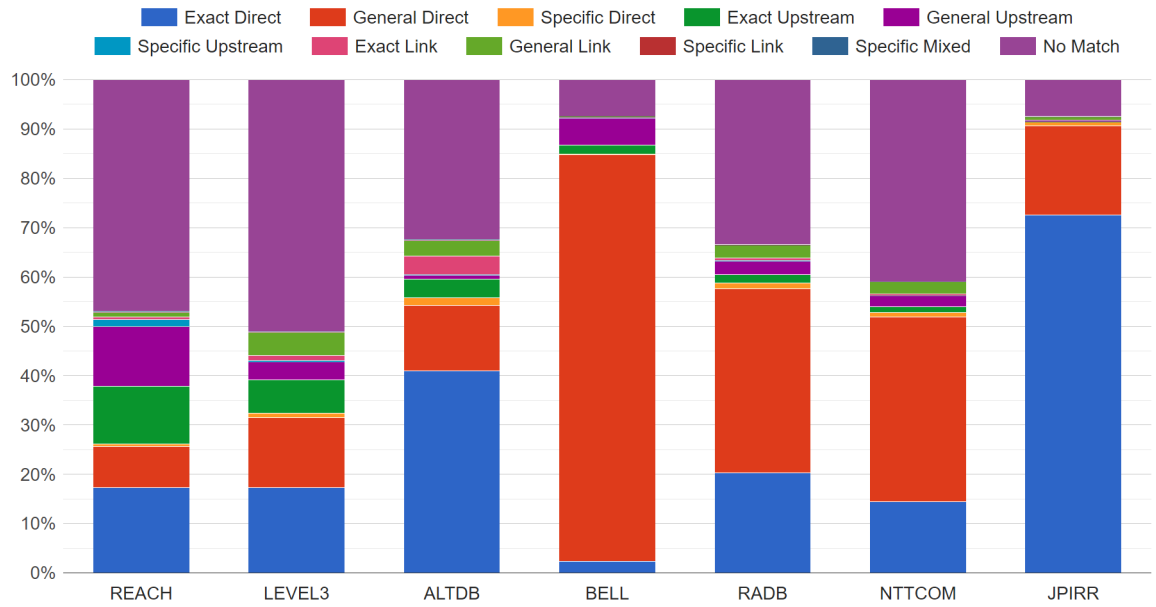


Figure 6.2: Route object matches for large routing registries

In Figure 6.2 we can see the generated matches for large routing registries that contain more than 10000 route objects. The largest routing registries are RADb, NTTCOM, and LEVEL3. Other route objects contain less than 50000 route objects. The percentage of route objects that generate no match is much larger when being compared with the routing registries of RIRs. Larger routing registries such as RADb, NTTCOM, and LEVEL3 contain more route objects that do not generate a match then smaller routing registries.

One reason why so many route objects return no match might be because of aggregation and re-origination. As an example, let us assume that a prefix 10.0.0.0/22 belongs to AS1 and AS1 is a provider of AS2. AS2 then is assigned 10.0.0.0/24 by AS1 for its own use and creates a route object (10.0.0.0/24, AS2). However, if AS1 announces to the rest of the Internet that it owns 10.0.0.0/22 and routes traffic destined for 10.0.0.0/24 internally to AS2, then the route object would only generate a Generic Link Match if it was known that AS1 is a provider of AS2. However, CAIDA's AS-relationship dataset might not always contain these relationships because it relies primarily on BGP data to detect these relationships. Routing registries of LEVEL3 and NTTCOM are primarily intended for customers and are used for automatic routing policy generation [17], it is mainly used for correct communication within the internal network of these providers.

Another explanation on why these routing registries have so many route objects that do not generate a match is that they contain an increased amount of outdated route objects that have not been removed. Outdated route objects often result in duplicate route objects with different origins for the same prefix. Table 6.1 shows us that registries that have a large number of routes that do not generate a match also have a high number of duplicates. After RGNET, routing registries with poor match results such as RADb, LEVEL3, NTTCOM, and REACH, have the highest amount of local duplicates. This indicates that the registries are more likely to contain outdated route objects that have not been removed. However, in some cases, a duplicate route object with a different origin is desired because the prefix is announced by multiple ASes. Because outdated route objects often do not impact routing as long as a valid route object also exists, the incentive to remove old route objects is often very low. It is however interesting to see that the average age of each route object shows is different between these registries. Whereas the route objects of NTTCOM, RADb, and REACH have an average age of 2014, 2013 and 2010 respectively, LEVEL3 has an average age of 2007. This means that many route objects in the LEVEL3 routing registry either do not change often or that it contains many outdated route objects.

Just as ARIN and APNIC, the BELL routing registry also shows a large percentage of Generic Direct Matches, showing an increased amount of aggregation. Among large routing registries, JPIRR has the highest amount of Exact Direct Matches and only 7.5% of the route objects generating no match.

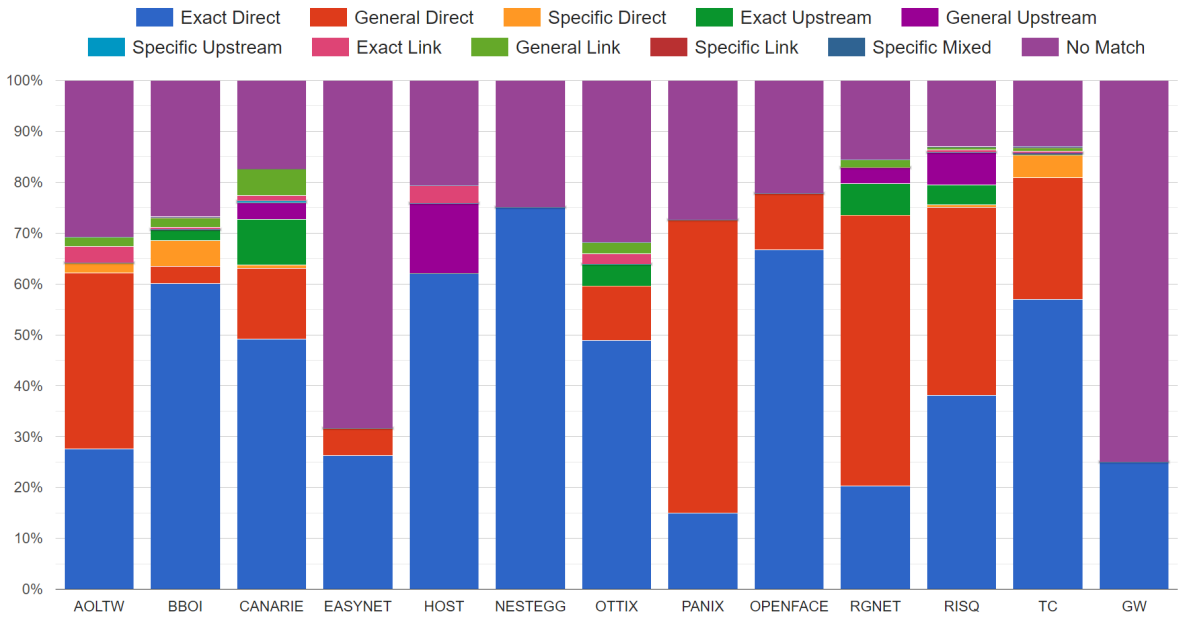


Figure 6.3: Route object matches for small routing registries

In Figure 6.3 the generated matches of route objects from small routing registries can be shown. Most routing registries such as NESTEGG, GW, and OPENFACE contain almost no route objects anymore (less than 20) but are included for completeness. All small routing registries other than TC, BBOI, CANARIE, and RISQ have less than 200 route objects and therefore only have a very small impact on the global accuracy of routing registries.

Most small routing registries have a generally large amount of route objects that do not generate a match. The larger routing registries of this category, such as TC, RISQ, and CANARIE have a larger percentage of route objects that generate a match. This is likely due to the fact that many network operators are transferring their route objects to larger routing registries such as ARIN, APNIC or RIPE. Network operators then forget to maintain or remove these route objects and instead focus on maintaining route objects in larger registries. EASYNET for example only contains 38 route objects of which almost 70% does not generate a match anymore. The average age of the route objects inside this routing registry is 2008 which shows that route objects are not often updated. We have also shown in Table 6.1 that 50% of these route objects is also contained in other registries. These indicators show that EASYNET is likely not actively maintained anymore.

6.1.2. Analyzing route objects from RIB and UPDATE perspective

In the previous section, we analyzed route objects from a registry perspective, trying to match all the route objects in different routing registries against prefixes present in RIBs. However, this way we cannot determine which routes contained in the RIB do not have an associated route object. Therefore we also compare all the prefixes inside a RIB against the union of all routing registries. This way we can determine how much percent of all the prefixes in the RIB is covered by the route objects present in the IRR. Since a RIB might store multiple routes for the same prefix (at most one for each peer), we perform the check on each route present in the RIB to make sure that different origins for a specific prefix due multi-homing or anycast are also covered.

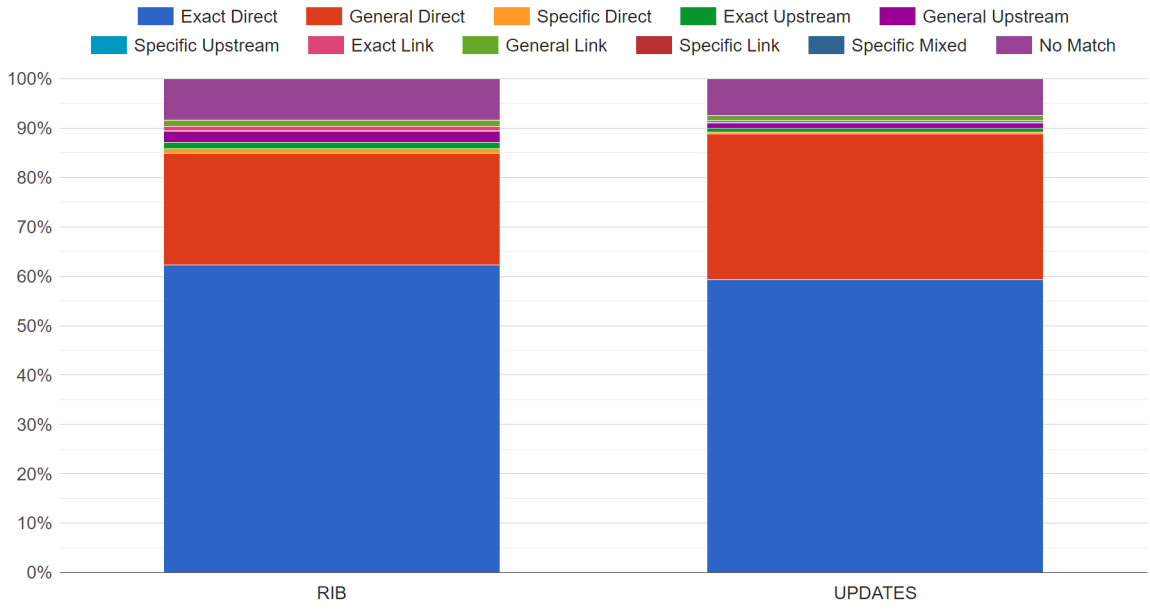


Figure 6.4: Route object matches from RIB perspective

As can be seen in Figure 6.4 roughly 85% of the route objects present in a RIB consist of either Exact Direct matches and General Direct matches. Together with all other match types, 92% of the prefixes inside a RIB can be matched. This basically means that roughly 8% of the prefixes is either not registered inside a routing registry or has an outdated route object. It is interesting to see that route objects are more often generating a match when processing UPDATEs compared to processing prefixes in a RIB. Because in 2018, knowledge about routing registries is more common, we suspect that organizations that want to announce a new prefix, are likely to create a new route object to increase acceptance by other ASes. RIBs also contain prefixes that were announced by smaller organizations a long time ago, at which routing registries were not yet widely known.

6.2. Detected Origin Hijacks

After running our detection algorithm on BGP announcements between 20 September 2018 and 18 October 2018 on RIPE RRC01, 902 origin hijacks were detected that were originating from 197 unique ASes. These origin hijacks are placed in three categories. A hijack is labeled as 'Bound' if it has a finite lifespan. Small hijacks of the same prefix and origin that appear in rapid succession will be grouped together as a single hijack with the duration of all the shorter hijacks combined. Hijacks are labeled 'Unbound' if they were still present in the RIB on the moment of analysis at 28 October 2018. A hijack is marked as 'AS_SET' if the origin it was announced from is an AS_SET. Announcing a prefix with an AS_SET as origin is considered a bad practice according to RFC6472 [46] because announcements cannot be validated due to the fact that the originator could be any AS from the AS_SET. Only the BGP speaker that aggregated these routes and the BGP speakers between the aggregator and the origin have knowledge about the true originator. Hijacks with an AS_SET as an origin attribute will be discussed in more detail in Section 7.1.1. The duration of each detected origin hijack was calculated using the RIPEStat BGP Update API [24]. This API provides a list of all the announcements and withdrawals of the prefix being queried. Using this data we can see when routes to the hijacker AS are withdrawn or overwritten by announcements containing another AS. Our algorithm was not executed on multiple RRC because of memory constraints (see Section 7.1.6) and the rate limitations imposed by LACNIC (see Section 7.1.3).

Detected Hijacks	902
- Bound duration	789
- Unbound duration	93
- AS_SET	20
Unique AS origins	197
- Only emitted bound hijacks	150
- Only emitted unbound hijacks	31
- Only emitted set hijacks	10
- Emitted multiple types	6
Unique AS hijackers per day	6
Detected hijacks per day	31

Table 6.2: Statistics on the Detected Origin Hijacks

As can be seen in Table 6.2, out of the 902 suspected origin hijacks, 789 were found to have a limited lifespan, 93 origin hijacks were still present on the moment of analysis. 20 prefixes were originated by an AS_SET origin and could therefore not be validated. Because our system is unable to validate prefixes that have an AS_SET as origin, we can conclude that during our detection period only 20 announcements were seen by RRC01 using an AS_SET as origin. The relatively low amount of announcements that use an AS_SET as origin also indicates that this is considered a bad practice. Even though multiple attempts were made to validate incoming announcements, it is still possible that an announcement that could not be validated is actually intended by the rightful owner. When an organization does not register the prefixes that it owns at a routing registry, does not use RPKI or does not have a direct owner relationship in RSEF listings, our detection system will regard it as an origin hijack whereas it might be intended behavior. It is however impossible to prove that our detected origin hijacks are really origin hijacks without contacting the involved operators, which would be very time-consuming. Origin hijacks often display unique characteristics which separate them from valid announcements. In Section 6.2.1 we will cover the duration of the detected origin hijacks with a bound duration. Because origin hijacks cause a loss in connectivity they are often detected and mitigated within a short amount of time. Detected origin hijacks with a short lifespan would, therefore, be more likely to be a misconfiguration or intended origin hijack. In Section 6.2.2 we analyze whether multiple of our detected origin hijacks appear at the same time. Because misconfiguration often triggers multiple origin hijacks simultaneously, more than 10 non-validated announcements announced by the same AS is a good indication of an origin hijack or route leak with re-origination. In Section 6.2.3 we manually inspect some of our detected cases to show typical cases of misconfiguration, or potentially intentional origin hijacks.

6.2.1. Duration of Bound Origin Hijacks

In order to gain more insight into the detected origin hijacks we analyze the duration of our detected origin hijacks with bound duration. As can be seen in Figure 6.9, 95% of the detected origin hijacks with a bound duration lasted less than 2.5 hours. 50% of the detected origin hijacks lasted less than 11 minutes. This indicates that most of the detected origin hijacks are resolved very quickly.

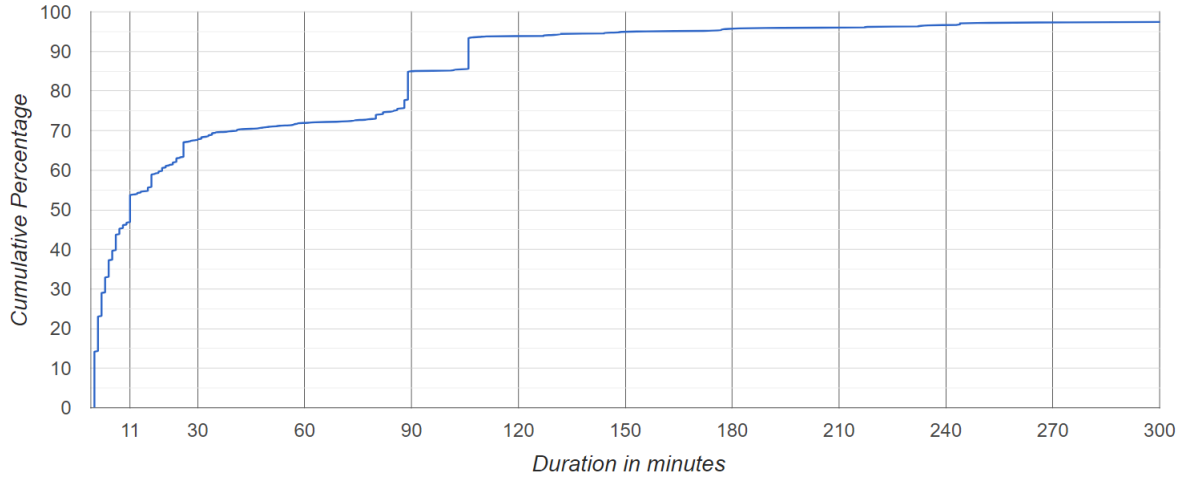


Figure 6.5: Duration of Detected Origin Hijacks with Finite Lifetime

If we take into account bound and unbound origin hijacks (excluding AS_SET announcements), 85% of the detected origin hijacks have a lifespan less then 2.5 hours. The average duration of bound origin hijacks was 11067 seconds or 184 minutes. The median duration of bound origin hijacks was 622 seconds or roughly 10 minutes. The minimum duration we found was 1 second and the maximum duration was 21 days. The reason why the average lifespan of all bound origin hijacks is relatively high compared to the median duration, is because 5% of the bound hijacks have durations that often last multiple days. If these suspected origin hijacks with long durations are, due unknown reasons, not present in the RIB at the moment of analysis it will be regarded as a bound origin hijack, whether the prefix hijack is likely to continue shortly afterward. We were not able to determine the reason why some of these suspected origin hijacks have small periods in which they are not present in the RIB.

Because 85% of all the announcements that could not be validated by our detection system lasted for less than 2.5 hours, it is likely to assume that these announcements are the results of misconfiguration or deliberate hijacking. The 11% of the detected origin hijacks that were unbound are either announcements by organizations that did not make any attempt at registering and protecting their own prefixes, or are origin hijacks that were not detected by the victim yet. When assigned prefixes, that are currently not being used, are hijacked, the victim might not detect this till the moment that they decide to actually use the assigned prefix. An example of such a case is the hijack of unused prefixes assigned to the Dutch Government by Romanian criminals [18], which was only mitigated after several days.

6.2.2. Analyzing Origin Hijacks from different perspectives

In order to gain more insight into the moment in time at which detected origin hijacks occurred we analyzed the amount of detected origin hijacks per day. Figure 6.6 shows the amount of detected origin hijacks per day. The maximum number of origin hijacks was detected on 2 October 2018 where 140 origin hijacks were detected, whereas the minimum number of 2 origin hijacks was on 13 October 2018.

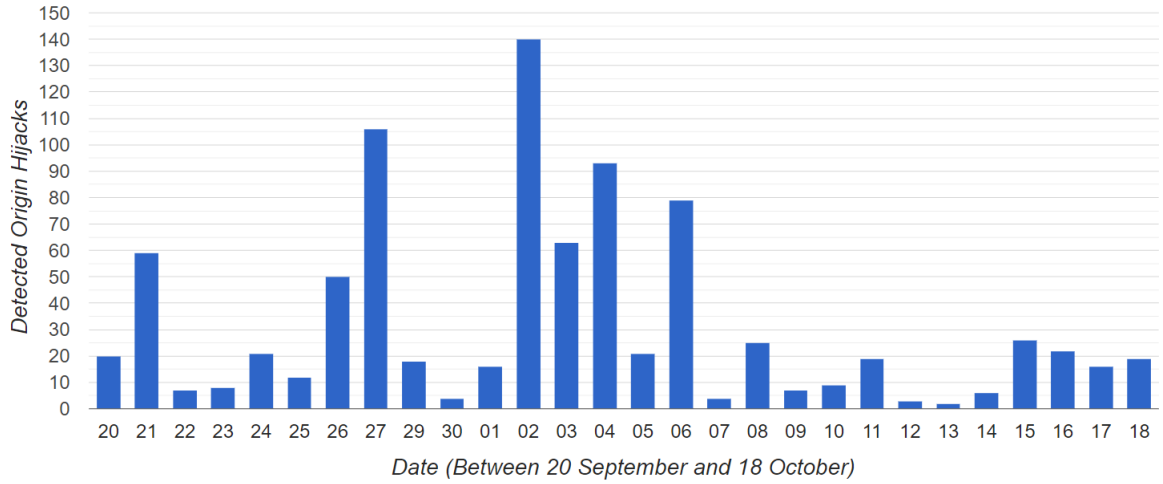


Figure 6.6: Duration of Detected Origin Hijacks with Finite Lifetime

When a network operator accidentally misconfigured a routing policy of a BGP speaker, it is often the case that multiple prefixes are hijacked at the same moment in time due to the prefixes being leaked and re-originated. When this happens, routes that should normally not be forwarded to a peer are now sent to the peer with the AS_PATH stripped off, making it appear as if the AS that leaks the routes originated the prefixes. AS_PATHs are often stripped off to reduce the path length on the global Internet. Shorter paths are more likely to attract traffic, and announcements with long paths are sometimes being filtered by some ASes. Figure 6.7 shows the hijacks per AS per day. Each colored bar presents the detected origin hijacks emitted by a single unique AS. Upon inspecting the figure we can see that a large amount of detected origin hijacks on 2 October is due to AS38733 hijacking 102 prefixes. The largest incidents in terms of the number of prefixes being hijacked were on 27 September (68 prefixes hijacked by AS30076), 2 October (102 prefixes hijacked by AS38733), 4 October (74 prefixes hijacked by AS6983) and 6 October (63 prefixes hijacked by AS45609) respectively. Section 6.2.3 will provide a case study on some of these incidents.

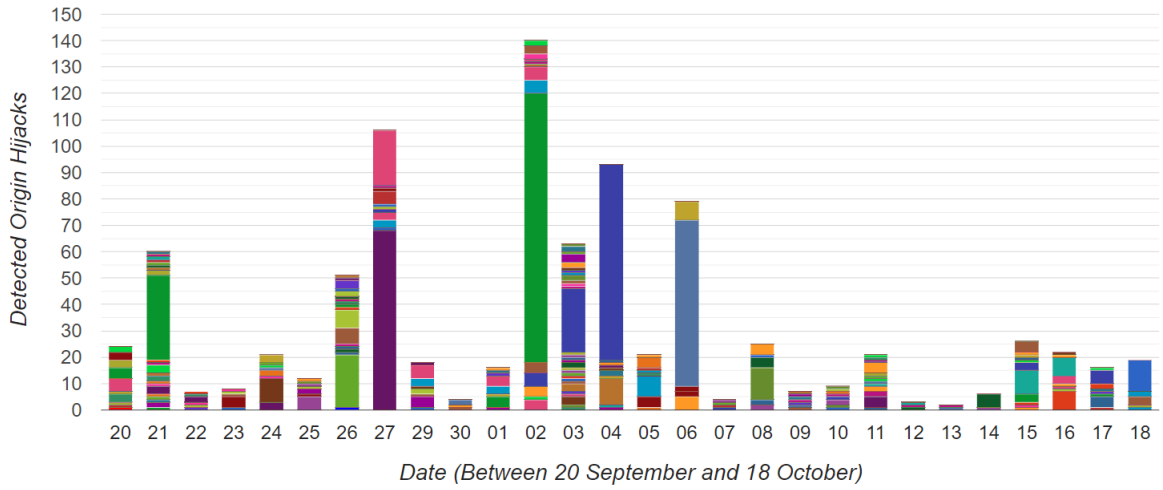


Figure 6.7: Bound origin hijack durations

Because one AS can have a very large impact on the total amount of origin hijacks on a single day as has been shown Figure 6.7, the amount of unique hijacking ASes might provide a better indication on the number of incidents that occur per day. Figure 6.8 shows the number of unique ASes that caused origin hijacks. The figure shows that on 3 October 29 unique hijacking ASes were detected whereas 2 October, which held the largest amount of suspected origin hijacks, only 14 unique hijackers were detected. Therefore one could state that more incidents were present on 3 October.

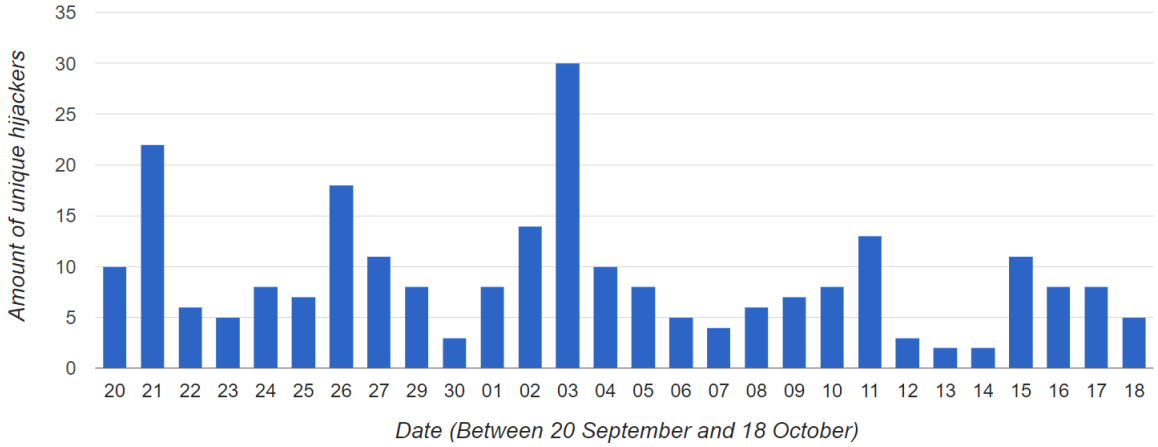


Figure 6.8: Bound origin hijack durations

In Figure 6.9 we can see the distribution of origin hijacks with a bound duration against those with an unbound duration. On 21 September a large number of origin hijacks were detected that were still present at the moment of analysis. On this day AS52055, belonging to *Mayak Smart Services*, announced 32 subprefixes of the 109.201.224.0/19 range with an unbound duration. We found that the original owner of these prefixes is a local internet registry named *Volia* located in Ukraine. We tried to contact them to verify these announcements but received no reply. We suspect that these prefixes were assigned by *Volia* to *Mayak Smart Services* but we were not able to verify this. Other detected origin hijacks with unbound duration were only announced in small groups of a maximum of 4 unbound origin hijacks per AS. Whether these origin hijacks are truly origin hijacks that were not yet mitigated, or prefixes that were not registered at any data source that our algorithm uses is unknown.

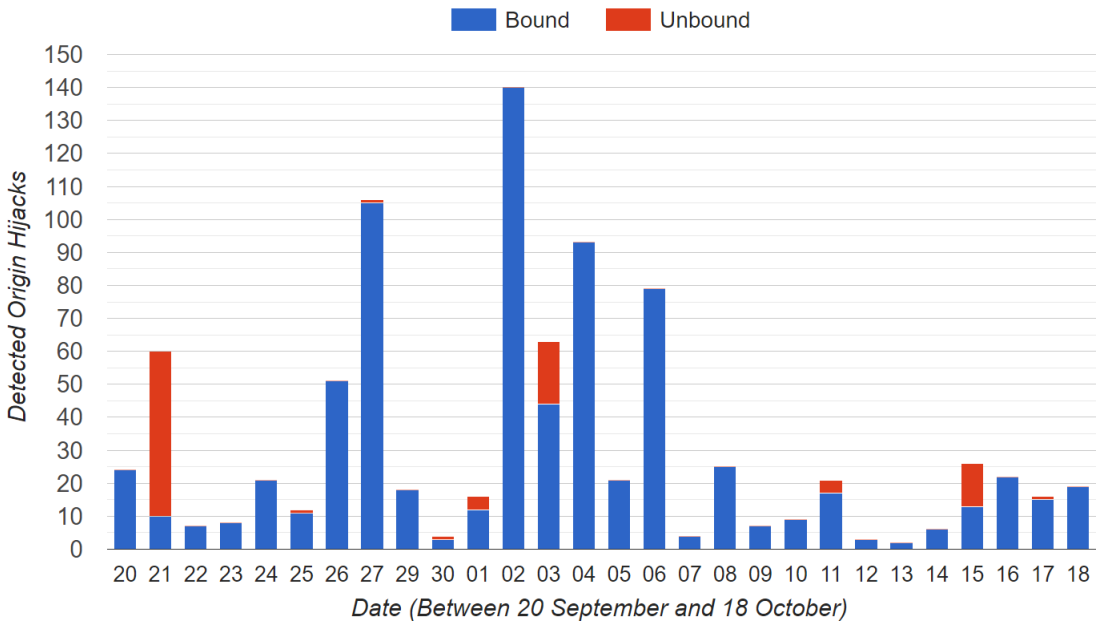


Figure 6.9: Bound origin hijack durations

6.2.3. Case Study

Case: Inland Northwest Health Services

On 27 October AS30076, registered as Inland Northwest Health Services (INHS), hijacked 68 prefixes mainly belonging China Telecom and APNIC. INHS is a non-profit organization in the US provides health care services. It also includes a division that is mainly concerned with TeleHealth. The hijack lasted for a duration of 10 minutes. The hijacked organizations and the prefixes that AS30076 owns are listed below.

Prefixes being hijacked by AS30076

- 32 prefixes of China Telecom Satellite (CTS)
- 7 prefixes of APNIC
- 3 prefixes of China Unicom
- 2 prefixes of KNET Technology (Beijing)
- 1 prefix of Beijing Founder Broadband Network Technology
- 1 prefix of China Education and Research Network
- 1 prefix of Beijing CheeryZone Scitech

Prefixes owned by AS30076

- 198.185.133.0/24
- 198.185.134.0/23
- 198.185.136.0/23

In Appendix A all hijacked prefixes are listed separately. The hijacked prefixes are all sub-prefixes from the 1.0.0.0/8 range which do not resemble the IP addresses that INHS already owned. A misconfiguration of so many prefixes is therefore unlikely.

On 21 October 2018, an academic article was published by the US Naval War College and Tel Aviv University stating that China Telecom Satellite (CTS), which is a state-owned telecommunications company, is hijacking the vital Internet backbone of western countries after the Xi-Obama agreement in 2015 [36]. The report shows that China Telecom has established 8 Points of Presence (PoPs) in the US and two in Canada since the beginning of the 2000s.

After inspection of the peers of AS30076, we found that AS30076 is peering with China Telecom in the US according to CAIDA's AS relationship dataset. Therefore a possible explanation would be that AS30076 leaked routes from China Telecom (AS19653) to Level3 (AS3356), stripping off the AS_PATH during the process, which in turn makes it appear as if AS30076 announced these prefixes. It is common to remove the AS_PATH and re-originate the announcement to shorten the AS_PATH seen on the global Internet. Shorter paths attract more traffic and announcements with long paths might be filtered by other ASes. When an export routing policy is misconfigured all prefixes learned from one peer (which in the case of a provider, might be a complete RIB table), will be sent to another neighbor. The reason why only the first 68 prefixes of such a RIB were hijacked is likely due to the fact that, when many new prefixes are suddenly being announced within a very short time frame, the receiving BGP speaker will block new incoming prefixes as a security measure. If measures like this were not in place, leaked routes that were re-originated could cause global outage within a short moment of time. Whether this route leak was intended by US state actors, obtaining short access to China Telecom's traffic, or an accidental incident will remain a mystery.

Case: Orange RDC

On 26 September 2018 at 10:31 AS37**84**3 hijacked 41.77.223.0/24. According to AFRINIC, AS37**84**3 is an reserved ASN and has not been assigned to a specific party yet. The following information could be retrieved from the detected origin hijacking report and by performing separate WHOIS queries.

Brief information:

- Hijacker ASN: AS37843
- Hijacker Organization: None, Reserved by AFRINIC
- Hijacked Prefix: 41.77.223.0/24
- Hijacked Prefix Owner: Orange RDC (formerly Congo Chine Telecom)
- AS_PATH: AS5511, AS37483, AS37483, AS37843
- Hijack Duration: 21 days

When we look at the AS_PATH we see that the direct upstream is AS37**48**3. AS37**48**3 is owned by Orange RDC, which is the same organization of the prefix that is being hijacked. AS37**48**3 also performs AS-prepending, adding its own AS twice to the AS_PATH. When we compare the origin AS and the upstream AS we see that they are very similar, by swapping the 8 and the 4 of the origin AS we can come up with the same AS as the upstream. Therefore it is very likely that this announcement was a misconfiguration by Orange RDC. Orange RDC likely intended to prepend its AS three times but made a small typo when specifying the ASes, resulting in AS37**84**3 as the origin of this announcement. The duration of this hijack lasted from 26 September 2018 till 27 October 2018, with small periods upon which the prefix was not being announced.

Case: Bharti Airtel Ltd. for GPRS Service

On 6 October 2018 at 16:31:54 AS45609 hijacked 63 prefixes. AS45609 belongs to Bharti Airtel Ltd. and is located in India. Bharti Airtel is a large global telecommunications company that operates in 16 countries at the moment of writing. When the origin hijack happened AS45609 announced 63 sub-prefixes of the 100.64.0.0/10 range which is a reserved Shared Address Space [65] that should not be globally routable.

Brief information:

- Hijacker ASN: AS45609
- Hijacker Organization: Bharti Airtel Ltd.
- Hijacked Prefixes: 63 subprefixes of the 100.64.0.0/10 range.
- Hijacked Prefix Owner: Reserved as Shared Address Space by IANA.
- Hijack Duration: 1 hour and 51 minutes
- AS_PATH: AS56730, AS31463, AS3223, AS9498, AS45609

The Shared Address Space is used for Carrier-Grade NAT (CGN) devices. In order for CGN to work it needs a routable IP address in order to communicate with customer networks. Private IP spaces (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) cannot be used for this because customer networks might already be using these IP addresses. If a prefix had to be assigned for each CGN the IPv4 address pool would have been depleted even faster. To avoid an even faster exhaustion of the IPv4 space IANA decided to reserve 100.64.0.0/10 for CGN networks.

The origin hijack lasted till 18:23:12 and had a total duration of 1 hour and 51 minutes. It is very likely that Bharti Airtel misconfigured a routing policy on one of their BGP speakers which caused their internal routes from the Shared Address Space to be announced to other ASes on the Internet as well. Because the Shared Address Space should not be globally routable, no specific parties were affected. It is however interesting to notice that the origin hijacks were able to reach the detecting RRC. Apparently, AS56730, AS31463, AS3223, and AS9498 do not block the propagation of announcements that include sub-prefixes of the reserved Shared Address Space, which could have prevented the origin hijack from reaching other ASes.

6.3. Comparison with BGPStream

In order to measure the performance of our system, our results are compared with those of BGPStream. Between 20 September 2018 and 18 October 2018, BGPStream detected 193 origin hijacks, whereas our system detected 902 origin hijacks. BGPStream was not able to detect all the origin hijacks that our system was able to. To verify this we checked if the cases presented in Section 6.2.3 were also present in the BGPStream dataset. We found that the Orange RDC and Bharti Airtel case were not detected by BGPStream, whereas both of them lasted for a significant amount of time. In order to compare the results of BGPStream with our own detection system each detected origin hijack by BGPStream was processed by our detection pipeline. Table 6.3 shows the results of this process.

Detected Hijacks (BGPStream)	193
Present in our dataset	36
Would have been detected earlier	84
Validated by WHOIS	27
Validated by RSEF	3
Validated by RPKI	0
Not seen by our RRC	43

Table 6.3: Comparison between BGPStream and our detected origin hijacks

As can be seen in Table 6.3 out of the 193 detected origin hijacks, only 36 were also present in our detected hijacks dataset. 84 of the detected origin hijacks could have been detected earlier in time by our system and were therefore not present in our detected hijacks. There were also cases that would have been validated at the WHOIS or RSEF phase, meaning that some of the origin hijacks present in BGPStream should not have been classified as origin hijacks at all. It is interesting to see that no origin hijacks were validated by RPKI which means that it is likely that BGPStream also uses RPKI ROAs to validate announcements. Upon manual inspection of detection hijacks at other moments in time we found that BGPStream does not use the routing registries proxied by LACNIC to validate origin hijacks, as some detected origin hijacks could have been validated directly by route objects present in these routing registries.

The reason why 43 of the detected origin hijacks of BGPStream are not present in our detected hijack set is due the fact that BGPStream uses many RRCs from BGPMon which are located at different ASes and might, therefore, detect different origin hijacks then our system does. Whereas BGPStream uses more than 100 different RRCs to detect origin hijacks, our results are based on the execution on a single RRC. If our system was executed on multiple RIPE RRCs the amount of hijacks seen only by BGPStream is likely to be much lower, based on the positioning of these RRCs within the Internet graph.

6.4. Performance

The performance of our detection system is largely determined by the performance of each individual filter. If a prefix is filtered away as soon as possible, we can avoid computing results for later filters. Therefore the order in which filters are present in our detection pipeline has great influence on the performance of the overall system. The outcome is not affected by the order of the filters, as a prefix is only marked as an origin hijack after it passed all the filters. To show the performance of each filter we benchmarked our detection system by processing 24 hours of UPDATE messages on 1 October 2018. The hardware being used for this benchmark is shown in Table 6.4. Only the CPU, memory and OS are listed here because no filter relies on disk usage. Table 6.5 shows the results per filter of our benchmark when running our detection system for 1 October 2018. Table 6.6 shows general statistics of our benchmark run and the individual performance of filters when applied to all the prefixes of 1 October 2018. The reason why the Route History and LACNIC filters are not included in these results is because these filters are too slow to compute on all prefixes of 1 October 2018.

Operating System	Debian 9 (Stretch)
CPU	Intel Core i5-4690 (4 cores)
Memory	8 Gb, 2400 MHz, Crucial Ballistix Tactical BLT2C4G3D1608ET3LX0CEU

Table 6.4: Benchmark setup

Filter	Prefixes processed	% of processed prefixes	Average time per prefix
RSEF	5431668	100%	13420 nanoseconds
MOAS and SOAS	310968	5.72509%	14140 nanoseconds
RPKI ROA	1556	0.02864%	38229 nanoseconds
WHOIS	1551	0.02855%	273007 nanoseconds
Route History	713	0.01312%	2.29 seconds
LACNIC	19	0.000003%	2.94 seconds

Table 6.5: Benchmark Results

		Individual Performance	% validated	Prefixes validated
Average duration per prefix	325204 nanoseconds	**MOAS and SOAS**	99.7699%	5419185
Throughput	3074 prefixes/second	**RSEF**	94.2748%	5120698
Total filter run-time	30 minutes	**WHOIS**	91.6184%	4976409
Total run-time	42 minutes	**RPKI**	11.5369%	626647

Table 6.6: Performance of our Detection System

As is shown in Table 6.5, we have chosen the RSEF filter as the first filter in our detection system due to two reasons. The first reason is that this filter needs relatively little time to assess the validity of a prefix. The second reason for placing the RSEF filter first is that it is able to validate a high percentage of prefixes. 94.2748% of the prefixes can be filtered out at this stage. This is the primary reason why our detection system is fast compared to other systems.

Afterward, the MOAS and SOAS and RPKI ROA filters follow, mainly because they are also very fast compared to other filters applied later in the detection pipeline. When applied on all the prefixes of 1 October 2018 the MOAS and SOAS filter is able to validate 99.7699% of the prefixes. The RSEF filter and the MOAS and SOAS filter combined were able to validate 99.97% of the prefixes only 77 seconds.

The RPKI ROA filter was only able to validate 5 prefixes. This is largely due to that fact that ROAs only cover roughly 11% of the prefixes in the world. It is however interesting to see that when measuring the RPKI filter individually it is able to validate 11.5% of the prefixes inside the UPDATE messages on 1 October 2018. The reason why only a few prefixes that reached the RPKI filter could be validated is unclear to us but we suspect that many routes that have an associated ROA also have a registered route object and where therefore validated earlier on.

The fourth filter that we picked is the WHOIS filter because it is the slowest filter that does not require sending network requests. Section 6.1.2 showed that a large percentage of prefixes could be validated. Table 6.6 also shows that when benchmarking the WHOIS filter individually on all the prefixes of 1 October 2018 roughly 92% of the prefixes could be validated, matching the accuracy shown in Section 6.1.2. It is interesting to see that when applying this filter on a long stream of UPDATE messages (as shown in Section 6.1.2) around 93% of the prefixes generates a match, whereas when applying it on all the prefixes that passed the RSEF, MOAS and SOAS and RPKI filters, only 46% could be validated. This is due to the fact that RSEF also uses route objects to filter out validated prefixes, which means that there is some overlap between the prefixes that are validated by RSEF and the prefixes that are validated by the WHOIS filter.

The primary bottleneck of our current system is Route History filter, which is tasked with checking whether this prefix or a parent prefix has already been announced by this origin before. Because we rely on the RIPEStat API to retrieve this information, a network request has to be made for each prefix that reaches this filter. It takes up 64% of the complete run-time duration and could, therefore, use some significant improvement in the future. How this can be done will be discussed in Section 7.1.2. The Route History filter is, however, able to filter out 97% of the prefixes, leaving only 19 prefixes for the LACNIC filter.

The reason why we separately query the LACNIC routing registry is that they do not provide a dump of their routing registry. They also apply a rate limit of 30 queries per 10 minutes. Therefore we want to limit our requests to the LACNIC routing registry as much as possible. By applying this filter as the last filter of our pipeline, we can limit the number of requests to a minimum.

The average duration per prefix is 325 microseconds which means that our detection system is able to process 3074 prefixes per second showing that it is easily able to handle multiple BGP streams if the system would have enough memory to load the corresponding data sources for each RRC. Even though the number of UPDATEs received depends on multiple factors such as the amount of peers and differences from day to day, our detection algorithm can be easily scaled to multiple systems in case a single system is not able to handle the desired amount of streams. This is due to the fact that our detection algorithm only needs to hold the state of the particular RRC that we are monitoring. The reason why there is a 12 minute difference between filter run-time and total run-time is mainly because data sources have to be loaded in memory. Downloading and parsing UPDATE messages also generate overhead into the system. Overhead will vary based on the network speed of the machine that the detection system is operating on.

7

Conclusion

We have presented a system for detecting origin hijacks using a filter-based approach. Chapter 3 has shown that the currently existing detection systems all have significant limitations that prevent them from detecting origin hijacks accurately and quickly.

Most other detection systems, such as [48], [66], [5], [68], [43], [61], [67] and [11], are not capable of detecting all origin hijacks that reach the Remote Route Collector because they make assumptions about origin hijacks that do not always hold. Because we are using a filter-based approach, only prefixes that could not be validated will be considered an origin hijack. Origin hijacks will therefore not be missed as long as the provided data sources do not contain erroneous data. This is an important improvement on existing systems because complex origin hijacks would otherwise go undetected. For example, the Orange RDC case which is described in Section 6.2.3 would not have been detected by solutions that use active probing such as [68] and [61], because a misconfiguration performed during AS prepending would not have resulted in a change in endpoint behavior. We have also shown in Section 6.2.3 that leaked routes that were re-originated can also be detected.

Detection systems such as [68] and [61], are not able to detect origin hijacks in real-time. If an origin hijack is detected earlier in time, a network operator can prevent potential damage from being done. Section 6.4 shows that our system is able to process 24 hours of UPDATE messages in 42 minutes, which proves that our system is capable of handling multiple BGP feeds at the same time. This is mainly due to the first filters in our detection pipeline being able to validate a high percentage of prefixes very quickly. Especially the RSEF filter and the MOAS and SOAS filter were able to validate 99.97% of the prefixes announced on 1 October 2018 in only 77 seconds.

Our detection system makes use of existing infrastructure, such as RPKI and routing registries, which contrasts solutions such as [41] and [68] that need new additional infrastructure, such as global databases, in order to work. The need for changes in infrastructure often significantly decreases the adoption rate of these technologies as is shown by, for example, BGPsec. Our detection system, on the other hand, can be directly deployed without the need for infrastructure changes. Our solution also increases in performance as the adoption of RPKI and routing registries grow.

When evaluating our solution in Chapter 6 we have shown that out of the 902 detected origin hijacks over a period of 29 days, 87% of the detected origin hijacks are shown to have a finite duration of which 95% had lasting less than 2.5 hours. This strongly suggests that these announcements were not desired and have been retracted after a short amount of time. This indicates that our system generates only a small amount of false positives, of which [43], [61], [67] and [11] were not found to capable. In Section 6.1 we have shown that routing registries are able to validate 92% of the prefixes inside a RIB. This contrasts the general belief that routing registries would not be accurate enough to be used for validation as has been stated by [55] and [60].

Unlike [66], [68], [43], [61] and [67], our system does not need to probe each prefix that was announced. Probing prefixes results in extra network overhead which is often undesired. Our system downloads the re-

quired data sources once each day, and only sends requests for a small portion of the prefixes that could not be validated earlier in our detection pipeline. Section 6.4 shows that only 732 network requests have to be sent per day, which is roughly 1 request every 2 minutes when spread over the entire day. Section 7.1.2 will propose a way to eliminate 713 of these requests, leaving only an average of 19 network requests per day when querying LACNIC. Section 7.1.3 explains a method to avoid making requests to LACNIC if it changes its infrastructure.

Our detection system improves on the shortcomings of existing solutions and introduced a new filter-based approach for validating origin hijacks. We have introduces new validation techniques such as filtering using RSEF listings and have proven the accuracy of controversial data sources such as routing registries and successfully used them for detecting origin hijacks. We have created new perspective on detecting origin hijacks by using a filter-based approach, which allows the detection of origin hijacks with great accuracy.

7.1. Discussion and Future Work

While constructing our detection system and performing our analysis we faced a variety of challenges. In this section, we will explain the challenges that we faced and the areas in which our detection system can be still be improved. Section 7.1.1 will cover announcements that use an AS_SET as origin and explains why we were not able to validate these announcements. Section 7.1.2 will cover the limitations of the current Routing History filter, described in Section 5.6, and discusses how this could be improved. Section 7.1.3 explains why we introduced a separate LACNIC filter in our detection pipeline and what changes to the LACNIC routing registry would be desired to enhance the performance of our detection system. Section 7.1.4 describes the challenge of keeping data sources up to date. Our current system operates on snapshots of data sources, opening the possibility of changes that might have been made to the data source between the moment that the snapshot has been taken and the moment that we use the snapshot when detecting origin hijacks. Section 7.1.5 discussing new filters that could be added to the detection system to reduce the number of false positives or add extra information which can be used when classifying the detected origin hijacks. Section 7.1.6 describes how memory usage of our current detection system can be reduced. Section 7.1.7 discusses running our current detection system on multiple RRCs. Section 7.1.8 proposes an analysis that can be done on routing registries that builds on the analysis that we have done in Section 6.1. Section 7.1.9 discusses the further classification of the detected origin hijacks in order to aid maintainers and network operators. It also discusses the issue of verification. Only the victim can know whether a detected origin hijack was truly intentional or not. In order to allow network operators to verify this, a separate system can be constructed.

7.1.1. AS_SET

Our current system is not able to validate origins that have an AS_SET as origin. This is due to the fact that any AS inside the AS_SET could be the announcing any part of the aggregated prefix. Only the AS that aggregated these prefixes into a larger prefix, creating an AS_SET in the process, and ASes between the true originator and the aggregating AS know which of the sub-prefixes is announced by who. As has been shown in Section 6.2 only 20 announcements with an AS_SET as its origin have been seen during a period of 29 days. RFC [46] has also stated that it is considered a bad practice because RPKI Route Origin Authorizations cannot be used on these announcements. In order for origin hijacks of their prefixes to be detected, network operators should stop announcing prefixes using an AS_SET as origin. Future announcements of their prefixes using an AS_SET as its origin could then immediately be considered suspicious.

7.1.2. Routing History Filter

As has been stated in Section 5.6, our current Routing History filter uses the RIPEStat Routing History API [24] to validate prefixes. This means that for each prefix reaching this filter, a separate network request has to be done. Section 6.4, showed that this results in the Routing History filter taking up 64% of the complete run-time duration, while it only has to process 0.01% of the total amount of prefixes. In order to improve upon this in the future, we could calculate the announcement history of each prefix by using the RIBs that are already provided by RIPE and Routeviews. Each unique origin by which a prefix has been announced in the past could then be stored completely memory. Because sending network requests would then not be necessary anymore, this filter would become significantly faster. The downside of this approach is that the system would require more memory and that computing the complete routing history in the past is time-consuming.

An estimation on the amount of memory that would be required can be calculated by multiplying the number of prefixes, we want to compute the routing history for, with the average amount of unique origins by which a prefix has been announced in the past. For example, if we have 750000 prefixes, which is the size of a typical RIB, that we want to compute the routing history for, and the average amount of origins in the past is 4, we would need at least $32 \cdot 4 \cdot 750000 = 96MB$. Depending on the data structure used for storing the history of these prefixes, more memory per prefix might be required.

In order to calculate the routing history per prefix, one has to parse through all the UPDATE messages that have been seen since the moment that an RRC had been created. This can be a time-consuming process but once the routing history per prefix has been constructed it only has to be maintained by adding new origins to the history when they occur. The RIPEStat Routing History API only uses RIB dumps to calculate the routing history for a certain prefix. Because of this, origin hijacks that would have been detected previously in time might be detected once more since the origin hijack has not been present in a RIB dump. This can be solved by using UPDATE messages to construct the routing history for each prefix.

Another limitation of the current Route History filter is that origin hijacks of the same prefix by the same origin that happened in the past, cannot be detected by our system. By allowing involved organizations of an origin hijack to verify whether an origin hijack was intended or not, we can remove origins from the routing history such that, if they would hijack the prefix again, would now be detected by our system again. If the Routing History filter would implement the described improvements, then the MOAS and SOAS filter would be redundant and could be removed from the detection pipeline. The prefixes that would be validated by the MOAS and SOAS filter would already be filtered by the improved Routing History filter efficiently. The potential improvements of the Routing History filter mentioned in this section are regarded as future work.

7.1.3. LACNIC

Currently, LACNIC is the only Regional Internet Registry that does not have its own routing registry but instead proxies the routing registries of NIC Brazil, NIC Chile, NIC Mexico. Because neither of these organizations provides a regular dump of their routing registry our current system is forced to send separate WHOIS query for each prefix, which in turn slows down our system. LACNIC also rate limits the number of queries that can be done to 30 queries per 5 minutes. In order to bypass the rate limit and the number of network requests as much as possible our current system queries LACNIC in the last filter of our detection pipeline. If these routing registries would expose a daily dump just as other routing registries do, our detection system could be significantly faster and could handle more BGP feeds at the same time. The LACNIC filter could then be integrated into the WHOIS filter, making it unnecessary as a separate filter.

7.1.4. Synchronizing data sources

Since our detection algorithm relies on a variety of data sources that are snapshots on a specific moment in time, a major challenge is keeping these data sources up to date. Because the data sources being used are all snapshots on a certain moment in time, changes might have occurred between the moment that the snapshot was taken and the moment an UPDATE that was received.

For example, if we use a routing registry dump from 1 October 2018 00:00:00 and try to match an announcement that was received at 6:00:00 at the same day, any associated route objects might have been changed during this time. If we were to detect origin hijacks in the present, this problem could be resolved by doing an explicit WHOIS query to the actual routing registry but this is not possible when detecting origin hijacks in the past. Sending explicit WHOIS queries also significantly slows down the system because extra network requests would have to be done. This problem also appears with other data sources such as RPKI dumps, RSEF listings and ASRank relationship datasets since they are also generated on a daily or monthly basis. This problem could be resolved if the data sources would be able to stream any changes made to these data sources in real-time. This way incremental changes can be applied on our snapshot to get the accurate state on a particular moment in time, avoiding the need to load new snapshots at specific moments in time.

7.1.5. Introduction of new filters

Our detection system is designed to be extensible. It is able to handle multiple BGP streams which can be processed in parallel, and new filters can easily be added. New filters can either reduce the number of detected origin hijacks by validating cases that our current system is not yet able to validate or to generate new

information about the detected origin hijacks which can be used when classifying the type of origin hijack that has been occurred. Examples of new filters that could be incorporated in a future version of our system would be a filter that checks whether reserved prefixes are announced or a filter that checks whether a prefix is on a blacklist because of spam activities. By adding new filters at a strategic location in the detection pipeline extra work only has to be done for a subset of the prefixes that reached these filter.

7.1.6. Memory Usage

The current implementation of our detection system can still use improvement in the amount of memory that is currently being used. RSEF listings and RPKI ROA dumps consist of only a few MBs, but the memory required when storing a RIB or all the route objects of routing registries is much larger. Currently our detection system stores all the routes of an entire RIB in memory. The memory consumed by the RIB of RRC01 consisted of 2.3 GB but can be even more when other RRCs with more peers are monitored. If the Routing History improvements discussed in Section 7.1.2 could be implemented the entire RIB would not be necessary anymore. But even if the Route History improvements would not be implemented we could still optimize this by storing only the unique origins per prefix. This would already greatly reduce the amount of memory being used because BGP path attributes do not have to be stored anymore. The routing registries dump also take up a significant amount of memory (1.3 GB) but can be shared between RRCs. We consider reducing the amount of memory that our system needs as future work.

7.1.7. Execution on multiple RRC

The results described in Section 6.2 were obtained from running our detection algorithm on a single RRC. This was done mainly because of the time and memory constraints that have been explained in Section 7.1.3 and Section 7.1.6. By running our detection algorithm on multiple RRC we would be able to detect more origin hijacks because some origin hijacks only have a limited scope. Section 6.3 has shown that we were not able to detect 43 origin hijacks that BGPStream was able to detect. By running our detection system on multiple RRCs we would be able to see how many origin hijacks that were not detected by our detection system would remain. This could give us insight on the coverage that the RRCs of BGPStream provide compared to the RRCs of RIPE and Routeviews.

7.1.8. Routing Registry Analysis

Section 6.1 showed the accuracy of route objects when being compared to prefixes inside a RIB. We have shown that the results vary significantly per routing registry. More research could be done on the differences per routing registry and why these differences occur. Further analysis of routing registries such as REACH, LEVEL3, RADB, and NTTCOM could verify whether the high percentage of route objects that do not generate a match is the result of a lack of maintenance or not. Other deviating results such as the high amounts of General Direct Matches in AFRINIC and ARIN-BULK should be researched more thoroughly in order to verify whether these are truly the result of aggregation by BGP speakers. We consider answering these kinds of questions as future work.

7.1.9. Further Classification and Verification

Currently, our detection system states whether an announcement is an origin hijack or not. The detection system could be improved by attempting to further classify the detected origin hijacks. Detected origin hijacks can then be classified as route leaks with re-origination, or mistyped prefixes for example. By further classifying the detected origin hijacks a network operator is able to understand the anomaly more quickly, enabling him to respond to incidents faster. The reports that are currently generated by our detection system contain a variety of information that can be used to make these classifications. Extra information related to the hijacker and potential victim should be looked up separately. We regard making these further classifications as future work. Another important aspect of detecting origin hijacks is that only the potential victim knows whether a detected origin hijack was intended or not. Even though our detection algorithm looks at many indicators that increase the likelihood of a hijack, the potential victim is the only one that can verify this. In order to verify whether an origin hijack was truly intended or not, a separate system could be constructed that allows network operators to confirm whether an attack was truly unintended or not. Automatic reports could be sent to the potential victims to make them aware of potential threats.

A

List of Routing Registries

This appendix will include a list of all the routing registries used in our analysis. It also includes the results for each routing registry when performing the route object analysis described in Section 6.1.

List of Routing Registries

- ALTDB
- AFRINIC
- AOLTW
- APNIC
- ARIN
- BELL
- BBOI
- CANARIE
- EASYNET
- RISQ
- NTTCOM
- RIPE
- ROGERS
- REACH
- PANIX
- RADB
- OTTIX
- OPENFACE
- NESTEGG
- JPIRR
- HOST
- TC
- GW
- LEVEL3
- WCGDB
- RGNET

Table A.1: Route object matches in routing registries on 1 September 2018

Registry	**Routes**	**EDM**	**GDM**	**SDM**	**EUM**	**GUM**	**SUM**	**ELM**	**GLM**	**SLM**	**SMM**	**No Match**
RADB	955316	194391	356013	10668	16387	26305	1378	5131	24200	389	736	319718
ARIN BULK	685651	19026	500789	1707	2795	5265	112	861	118088	24	127	36857
NTTCOM	380966	54895	142579	3447	4860	8025	360	1483	9175	160	172	155810
RIPE	355457	175081	110908	4887	7992	5026	206	2273	7598	68	278	41140
LACNIC	221074	17534	130001	6594	310	1811	33	193	1805	8	320	62465
APNIC	197516	44756	60325	3762	927	2585	50	1497	1863	81	122	81548
LEVEL3	127425	22041	18141	1132	8645	4517	411	1334	5954	58	54	65138
ARIN IRR	39736	13813	10473	595	624	779	28	256	1707	9	31	11421
REACH	32809	5666	2758	166	3820	3992	471	159	302	16	57	15402
BELL	29468	693	24309	19	535	1620	28	25	57	0	5	2177
ALTDB	14127	5783	1886	209	541	107	18	530	445	12	26	4570
AFRINIC	10500	3163	6755	123	23	77	1	3	1	0	2	352
JPIRR	10283	7457	1861	78	12	25	5	4	66	1	7	767
TC	5973	3400	1435	261	10	12	0	25	40	2	13	775
BBOI	3328	1999	112	169	68	4	2	13	63	1	4	893
CANARIE	1448	712	201	10	130	47	5	15	75	0	1	252
RISQ	903	344	334	4	36	57	0	5	4	0	1	118
AOLTW	156	43	54	3	0	0	0	5	3	0	0	48
RGNET	64	13	34	0	4	2	0	0	1	0	0	10
OTTIX	47	23	5	0	2	0	0	1	1	0	0	15
PANIX	40	6	23	0	0	0	0	0	0	0	0	11
EASYNET	38	10	2	0	0	0	0	0	0	0	0	26
HOST	29	18	0	0	0	4	0	1	0	0	0	6
OPENFACE	18	12	2	0	0	0	0	0	0	0	0	4
NESTEGG	4	3	0	0	0	0	0	0	0	0	0	1
Total	3072376	570882	1369000	33834	47721	60260	3108	13814	171448	829	1956	799524
Percentage	100%	18,58%	44,55%	1,10%	1,55%	1,96%	0,10%	0,45%	5,58%	0,03%	0,06%	26,02%

EDM: Exact Direct Match, GDM: General Direct Match, SDM: Specific Direct Match, EUM: Exact Upstream Match, GUM: General Upstream Match, SUM: Specific Upstream Match, ELM: Exact Link Match, GLM: General Link Match, SLM: Specific Link Match, SMM: Specific Mixed Match.
For more details see Section 5.5.

Table A.2: Route object matches in routing registries on 1 September 2018

Type	**Routes**	**EDM**	**GDM**	**SDM**	**EUM**	**GUM**	**SUM**	**ELM**	**GLM**	**SLM**	**SMM**	**No Match**
RIB	305745884	190507676	69153819	2743872	3967354	6790656	260195	3022914	3314014	898773	99012	24987599
Percentage	100%	62.31%	22.61%	0.89%	1.29%	2.22%	0.05%	0.98%	1.08%	0.29%	0.03%	8.17%
UPDATES	10718272	6354480	3167257	44014	73386	120039	8037	46519	96213	5304	2255	800768
Percentage	100%	59.28%	29.55%	0.41%	0.68%	1.11%	0.07%	0.43%	0.89%	0.04%	0.02%	7.47%

EDM: Exact Direct Match, GDM: General Direct Match, SDM: Specific Direct Match, EUM: Exact Upstream Match, GUM: General Upstream Match, SUM: Specific Upstream Match, ELM: Exact Link Match, GLM: General Link Match, SLM: Specific Link Match, SMM: Specific Mixed Match.
For more details see Section 5.5.

A

Origin Hijack Cases

This appendix will list the details of each of the cases presented in Section 6.2.3.

A.1. Orange RDC

Information in this section supplements the case study on *Orange RDC* written in Section 6.2.3.

Brief information:

- Hijacker ASN: AS37843
- Hijacker Organization: None, Reserved by AFRINIC
- Hijacked Prefix: 41.77.223.0/24
- Hijacked Prefix Owner: Orange RDC (formerly Congo Chine Telecom)
- AS_PATH: AS5511, AS37483, AS37483, AS37843
- Hijack Duration: 21 days

A.2. Bharti Airtel Ltd. for GPRS Service

Information in this section supplements the case study on *Bharti Airtel Ltd for GPRS Service* written in Section 6.2.3.

Brief information:

- Hijacker ASN: AS45609
- Hijacker Organization: Bharti Airtel Ltd.
- Hijacked Prefixes: 63 subprefixes of the 100.64.0.0/10 range.
- Hijacked Prefix Owner: Reserved as Shared Address Space by IANA.
- Hijack Duration: 1 hour and 51 minutes
- AS_PATH: AS56730, AS31463, AS3223, AS9498, AS45609

List of prefixes hijacked by AS45609

- 100.64.0.0/17
- 100.64.128.0/17
- 100.65.0.0/17
- 100.65.128.0/17
- 100.66.0.0/17
- 100.66.128.0/17
- 100.67.0.0/17
- 100.67.128.0/17
- 100.68.0.0/17
- 100.68.128.0/17
- 100.69.0.0/17
- 100.69.128.0/17
- 100.70.0.0/17
- 100.70.128.0/17
- 100.71.0.0/17
- 100.71.128.0/17
- 100.76.0.0/17
- 100.76.128.0/17
- 100.77.0.0/17
- 100.77.128.0/17
- 100.78.0.0/17
- 100.78.128.0/17
- 100.79.0.0/17
- 100.79.128.0/17
- 100.80.0.0/17
- 100.80.128.0/17
- 100.81.0.0/17
- 100.81.128.0/17
- 100.82.0.0/17
- 100.82.128.0/17
- 100.83.0.0/17
- 100.83.128.0/17
- 100.88.0.0/17
- 100.88.128.0/17
- 100.89.0.0/17
- 100.89.128.0/17
- 100.90.0.0/17
- 100.90.128.0/17
- 100.91.0.0/17
- 100.91.128.0/17
- 100.92.0.0/17
- 100.92.128.0/17
- 100.93.0.0/17
- 100.93.128.0/17
- 100.94.0.0/17
- 100.94.128.0/17
- 100.95.0.0/17
- 100.95.128.0/17
- 100.100.0.0/18
- 100.100.128.0/18
- 100.100.192.0/18
- 100.100.64.0/18
- 100.101.0.0/18
- 100.101.128.0/18
- 100.101.192.0/18
- 100.101.64.0/18
- 100.102.0.0/18
- 100.102.128.0/18
- 100.102.192.0/18
- 100.103.0.0/18
- 100.103.128.0/18
- 100.103.192.0/18
- 100.103.64.0/18

A.3. Inland Northwest Health Services

Information in this section supplements the case study on *Inland Northwest Health Services* (AS30076) written in Section 6.2.3. This hijack occurred for 10 minutes.

Complete list of hijacked prefixes by AS30076

- 1.0.1.0/24: China Telecom (CN)
- 1.0.2.0/23: China Telecom (CN)
- 1.0.32.0/19: China Telecom (CN)
- 1.0.8.0/21: China Telecom (CN)
- 1.1.0.0/24: China Telecom (CN)
- 1.1.16.0/20: China Telecom (CN)
- 1.1.2.0/23: China Telecom (CN)
- 1.1.32.0/19: China Telecom (CN)
- 1.1.4.0/22: China Telecom (CN)
- 1.1.8.0/21: Asia Pacific Network Information Centre (AU)
- 1.10.0.0/21: China Telecom (CN)
- 1.10.11.0/24: China Telecom (CN)
- 1.10.12.0/22: China Telecom (CN)
- 1.10.16.0/20: China Telecom (CN)
- 1.10.32.0/19: China Telecom (CN)
- 1.10.64.0/18: China Telecom (CN)
- 1.10.8.0/23: China Telecom (CN)
- 1.116.0.0/14: Asia Pacific Network Information Centre (AU)
- 1.12.0.0/14: Beijing Founder Broadband Network Technology Co.,Ltd (CN)
- 1.180.0.0/14: China Telecom (CN)
- 1.184.0.0/15: China Education and Research Network Guangzhou Regional Network (CN)
- 1.188.0.0/14: China Unicom (CN)
- 1.192.0.0/13: China Telecom (CN)
- 1.2.0.0/23: China Telecom (CN)
- 1.2.16.0/20: China Telecom (CN)
- 1.2.2.0/24: KNET Techonlogy (BeiJing) Co.,Ltd. (CN)
- 1.2.32.0/19: China Telecom (CN)
- 1.2.4.0/22: Asia Pacific Network Information Centre (AU)
- 1.2.64.0/18: China Telecom (CN)
- 1.2.8.0/21: Asia Pacific Network Information Centre (AU)
- 1.202.0.0/15: China Telecomk (CN)
- 1.204.0.0/14: China Telecom (CN)
- 1.24.0.0/13: China Unicom (CN)
- 1.3.0.0/16: China Telecom (CN)
- 1.4.1.0/24: China Telecom (CN)
- 1.4.16.0/20: China Telecom (CN)
- 1.4.2.0/23: China Telecom (CN)
- 1.4.32.0/19: China Telecom (CN)
- 1.4.4.0/22: Asia Pacific Network Information Centre (AU)
- 1.4.64.0/18: China Telecom (CN)
- 1.4.8.0/21: China Telecom (CN)
- 1.45.0.0/16: Beijing CheeryZone Scitech Co.,Ltd (CN)
- 1.48.0.0/14: Asia Pacific Network Information Centre (AU)
- 1.56.0.0/13: China Unicom (CN)
- 1.68.0.0/14: China Telecom (CN)
- 1.8.0.0/16: KNET Techonlogy (BeiJing) Co.,Ltd. (CN)
- 1.80.0.0/12: Asia Pacific Network Information Centre (AU)
- 14.0.0.0/21: China Telecom (CN)
- 14.0.12.0/22: China Telecom (CN)
- 14.1.0.0/22: China Telecom (CN)
- 14.1.108.0/22: Shanghai Yunrui Zhitong Industrial Co. Ltd. (CN)
- 14.1.24.0/22: Beijing Teletron Telecom Engineering Co., Ltd. (CN)
- 14.1.96.0/22: Nanning Youmi Network Technology Co., Ltd. (CN)
- 14.102.128.0/22: China Telecom (CN)
- 14.102.156.0/22: 1065-A Zhaojiabang Road (CN)
- 14.102.180.0/22: Beijing Internet Harbor Technology Co.,Ltd (CN)
- 14.103.0.0/16: Beijing Bitone United Networks Technology Service Co.,Ltd. (CN)
- 14.104.0.0/13: China Telecom (CN)
- 14.112.0.0/12: China Telecom (CN)
- 14.130.0.0/15: BeiJing New-Billion Telecom Technology Co.,Ltd (CN)
- 14.134.0.0/15: China Telecom (CN)
- 14.144.0.0/12: China Telecom (CN)
- 14.16.0.0/12: China Telecom (CN)
- 14.192.60.0/22: Beijing Bluetomedia Co.LTD. (CN)
- 14.192.76.0/22: China Telecom (CN)
- 14.196.0.0/15: Beijing Weishichuangjie Technical Development Co.,Ltd (CN)
- 14.204.0.0/15: China Unicom (CN)
- 14.208.0.0/12: China Telecom (CN)

Bibliography

[1] APNIC - Measuring the Adoption of RPKI Route Origin Validation
https://blog.apnic.net/2018/06/19/measuring-the-adoption-of-rpki-route-origin-validation/

[2] APNIC WHOIS Guide
https://www.apnic.net/manage-ip/using-whois/guide/route/

[3] ARIN Internet Routing Registry Roadmap
https://www.arin.net/vault/resources/routing/2018_roadmap.html

[4] ARIN - Membership Regulations
https://www.arin.net/public/aboutARINonline.xhtml

[5] ARTEMIS: Neutralizing BGP Hijacking within a Minute
https://arxiv.org/pdf/1801.01085.pdf

[6] CAIDA ASRank
http://as-rank.caida.org/

[7] ThousandEyes - Route Leak causes Amazon and AWS outage
https://blog.thousandeyes.com/route-leak-causes-amazon-and-aws-outage

[8] DYN Blog - Shutting down the BGP Hijack Factory
https://dyn.com/blog/shutting-down-the-bgp-hijack-factory

[9] NIST RIB Estimation
http://www.nist.gov/itl/antd/upload/BGPSEC_RIB_Estimation.pdf

[10] IANA BGP Capability Codes
https://www.iana.org/assignments/capability-codes/capability-codes.xhtml

[11] Detection of BGP Hijacking using TTL Analysis
http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2018/MSC/MSC-2018-05.pdf

[12] CAIDA AS-relationship dataset
http://data.caida.org/datasets/as-relationships/

[13] APNIC - Analysis of the DODO Telsta incident
https://labs.apnic.net/blabs/?p=139/

[14] The Internet Routing Registry
www.irr.net

[15] NANOG 41 - Detecting Routing Leaks by Counting
https://www.nanog.org/meetings/nanog41/presentations/mauch-lightning.pdf

[16] NIST RPKI Monitor
https://rpki-monitor.antd.nist.gov/

[17] NTTCOM - Membership Regulations
https://www.us.ntt.net/support/policy/rr-faq.cfm

[18] BuZa Incident
https://tweakers.net/nieuws/104975/ip-adressen-buza-gekaapt-via-bgp-hijacking.html

[19] Dyn - Pakistan Hijack Incident
https://dyn.com/blog/pakistan-hijacks-youtube-1/

[20] RADb - Membership Regulations
https://www.radb.net/register/

[21] RIPE - Managing Route Objects in the IRR
https://www.ripe.net/manage-ips-and-asns/db/support/managing-route-objects-in-the-irr

[22] RIPE Route Information Service
https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris

[23] RIPE RSEF Guide
https://ftp.ripe.net/pub/stats/ripencc/RIR-Statistics-Exchange-Format.txt

[24] RIPEStat API
https://stat.ripe.net/docs/data_api#RoutingHistory

[25] Regional Internet Registry Allocation Statistics
http://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/RIPENCC/IPv4-ByNb.html

[26] AFRINIC RPKI Validator
http://validator.afrinic.net:8080/bgp-preview

[27] Bahaa Al-Musawi, Philip Branch, and Grenville Armitage. BGP Anomaly Detection Techniques - A Survey

[28] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. Routing Policy Specification Language (RPSL). RFC 2622, RFC Editor, June 1999

[29] T. Bates, E. Gerich, L. Joncheray, J-M. Jouanigot, D. Karrenberg, M. Terpstra, and J. Yu. Representation of IP Routing Policies in a Routing Registry (ripe-81++). RFC 1786, RFC Editor, March 1995

[30] T. Bates, R. Chandra, D. Katz, and Y. Rekhter. Multiprotocol Extensions for BGP-4. RFC 4760, RFC Editor, January 2007

[31] Tomasz Bilski. Disaster's Impact on Internet Performance – Case Study. In Andrzej Kwiecie, Piotr Gaj, and Piotr Stera, editors, *Computer Networks*, pages 210–217, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-02671-3

[32] L. Blunk, M. Karir, and C. Labovitz. Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format. RFC 6396, RFC Editor, October 2011

[33] R. Chandra, P. Traina, and T. Li. BGP Communities Attribute. RFC 1997, RFC Editor, August 1996

[34] L. Daigle. WHOIS Protocol Specification. RFC 3912, RFC Editor, September 2004

[35] I. O. de Urbina Cazenave, E. Köşlük, and M. C. Ganiz. An anomaly detection framework for BGP. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 107–111, June 2011

[36] Chris Demchak, , and Yuval Shavitt and. China's Maxim – Leave No Access Point Unexploited: The Hidden Story of China Telecom's BGP Hijacking. *Military Cyber Affairs*, 3(1), jun 2018

[37] S. Deshpande, M. Thottan, T. K. Ho, and B. Sikdar. An Online Mechanism for BGP Instability Detection and Analysis. *IEEE Transactions on Computers*, 58(11):1470–1484, Nov 2009. ISSN 0018-9340

[38] Xenofontas Dimitropoulos and George Riley. Modeling Autonomous-System Relationships. In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation*, PADS '06, pages 143–149, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2587-3

[39] Lixin Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, Dec 2001. ISSN 1063-6692

[40] Vasileios Giotsas and Shi Zhou. Inferring Internet AS Relationships Based on BGP Routing Policies. 06 2011

[41] Andreas Haeberlen, Ioannis Avramopoulos, Jennifer Rexford, and Peter Druschel. NetReview: Detecting When Interdomain Routing Goes Wrong., 01 2009

[42] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). BCP 1930, RFC Editor, March 1996

[43] X. Hu and Z. M. Mao. Accurate Real-time Identification of IP Prefix Hijacking. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 3–17, May 2007

[44] E. Jasinska, N. Hilliard, R. Raszuk, and N. Bakker. Internet Exchange BGP Route Server. RFC 7947, RFC Editor, September 2016

[45] J. Karlin, S. Forrest, and J. Rexford. Pretty Good BGP: Improving BGP by Cautiously Adopting Routes. In *Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 290–299, Nov 2006

[46] W. Kumari and K. Sriram. Recommendation for Not Using AS_SET and AS_CONFED_SET in BGP. BCP 172, RFC Editor, December 2011

[47] Mohit Lad, Xiaoliang Zhao, Beichuan Zhang, Daniel Massey, and Lixia Zhang. Analysis of BGP Update Surge during Slammer Worm Attack. In *IWDC*, 2003

[48] Mohit Lad, Daniel Massey, Dan Pei, Yiguo Wu, Beichuan Zhang, and Lixia Zhang. PHAS: A Prefix Hijack Alert System. In *USENIX Security Symposium*, 2006

[49] M. Lepinski and K. Sriram. BGPsec Protocol Specification. RFC 8205, RFC Editor, September 2017

[50] M. Lepinski, S. Kent, and D. Kong. A Profile for Route Origin Authorizations (ROAs). RFC 6482, RFC Editor, February 2012

[51] Jun Li, Dejing Dou, Zhen Wu, Shiwoong Kim, and Vikash Agarwal. An internet routing forensics framework for discovering rules of abnormal BGP events. *Computer Communication Review*, 35:55–66, 2005

[52] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. AS Relationships, Customer Cones, and Validation. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 243–256, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1953-9

[53] A. Lutu, M. Bagnulo, J. Cid-Sueiro, and O. Maennel. Separating wheat from chaff: Winnowing unintended prefixes using machine learning. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 943–951, April 2014

[54] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding BGP Misconfiguration. *ACM SIGCOMM Computer Communication Review*, 32, 09 2003

[55] D. McPherson, S. Amante, E. Osterweil, L. Blunk, and D. Mitchell. Considerations for Internet Routing Registries (IRRs) and Routing Policy Configuration. RFC 7682, RFC Editor, December 2015

[56] Bahaa Musawi. Detecting BGP Anomalies Using Recurrence Quantification Analysis, 03 2018

[57] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, January 2006

[58] K. Sriram, D. Montgomery, D. McPherson, E. Osterweil, and B. Dickson. Problem Definition and Classification of BGP Route Leaks. RFC 7908, RFC Editor, June 2016

[59] Kotikalapudi Sriram, Doug Montgomery, Brian Dickson, Keyur Patel, and Andrei Robachevsky. Methods for Detection and Mitigation of BGP Route Leaks. Internet-Draft draft-ietf-idr-route-leak-detection-mitigation-08, IETF Secretariat, March 2018

[60] Richard Steenbergen. NANOG 44 - What is wrong with IRR. `https://www.youtube.com/watch?v=39bA6feM5uA`

[61] M. Tahara, N. Tateishi, T. Oimatsu, and S. Majima. A method todetect prefix hijacking by using ping tests. page 390–398, 2008. ISSN 1542-1201

[62] Georgios Theodoridis, Orestis Tsigkas, and Dimitrios Tzovaras. A Novel Unsupervised Method for Securing BGP Against Routing Hijacks. In *ISCIS*, 2012

[63] Q. Vohra and E. Chen. BGP Support for Four-Octet Autonomous System (AS) Number Space. RFC 6793, RFC Editor, December 2012

[64] Matthias Wählisch, Olaf Maennel, and Thomas C. Schmidt. Towards Detecting BGP Route Hijacking Using the RPKI. *SIGCOMM Comput. Commun. Rev.*, 42(4):103–104, August 2012. ISSN 0146-4833

[65] J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, and M. Azinger. IANA-Reserved IPv4 Prefix for Shared Address Space. BCP 153, RFC Editor, April 2012

[66] Y. Xiang, Z. Wang, X. Yin, and J. Wu. Argus: An accurate and agile system to detecting IP prefix hijacking. In *2011 19th IEEE International Conference on Network Protocols*, pages 43–48, Oct 2011

[67] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush. iSPY: Detecting IP Prefix Hijacking on My Own. *IEEE/ACM Transactions on Networking*, 18(6):1815–1828, Dec 2010. ISSN 1063-6692

[68] Changxi Zheng, Lusheng Ji, Dan Pei, Jia Wang, and Paul Francis. A light-weight distributed scheme for detecting ip prefix hijacks in real-time. In *SIGCOMM*, 2007